Obsah

Úvod a plánování

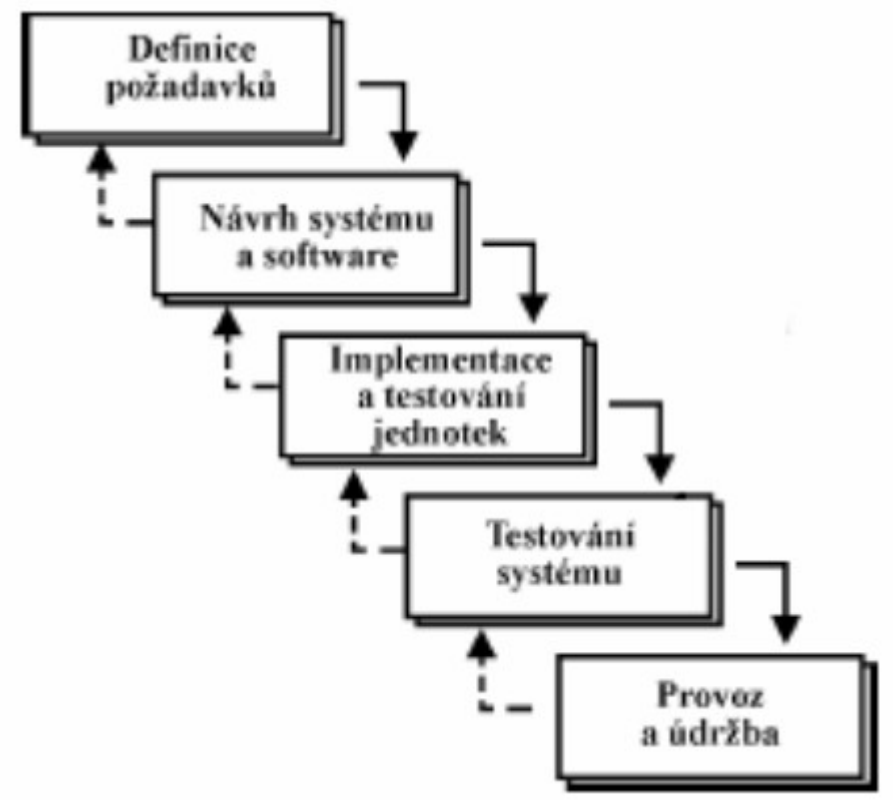
Model životního cyklu vodopád

Vodopádový model je sekvenční vývojový proces, ve kterém je vývoj nahlížen jako neustále se svažující tok (jako když teče vodopád) fázemi analýzy požadavků, návrhu, implementace, testování (validace), integrace a údržby. Jako první formální popis vodopádového modelu je často citován článek který publikoval v roce 1970 Winston W. Royce (1929–1995),[3] ačkoli Royce pojem „vodopádový“ ve svém článku nepoužil. Royce paradoxně představil tento model jako příklad chybného, nefungujícího modelu.[4] Hlavními principy vodopádového přístupu[2]:

Projekt je rozdělen na fáze jdoucí postupně za sebou, přičemž některé se mohou překrývat.

Důraz je kladen na plánování, časové rozvrhy, termíny, rozpočty a realizace celého systému najednou.

Přísná kontrola je udržována po celou dobu životnosti projektu prostřednictvím využití rozsáhlých písemných dokumentů, jakož i prostřednictvím formálních revizí a schvalování uživatelem (signoff) a na konci většiny fází a vstupy od managementu informačních technologií před začátkem další fáze.



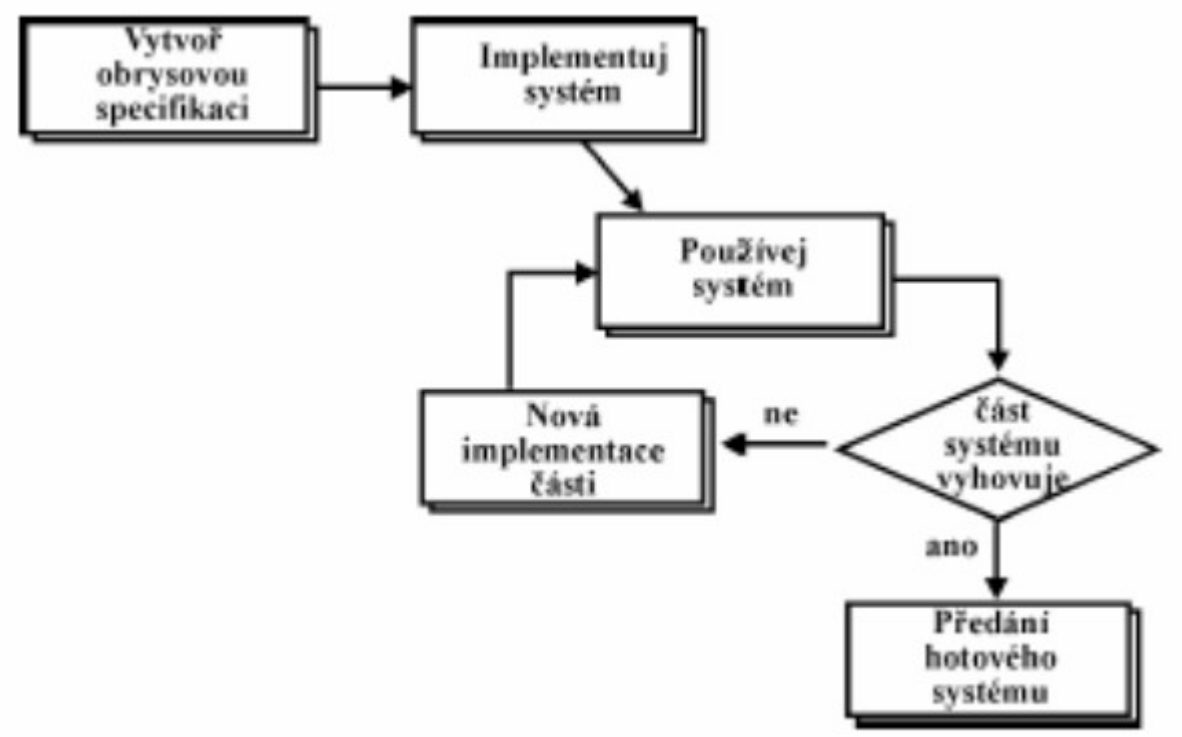
Inkrementální (přírůstkový) přístup

Inkrementální přístup je vhodný pro kombinaci sekvenčních a iteračních metodik softwarového vývoje. Cílem je omezit projektová rizika rozdělením projektu **na menší segmenty** a zjednodušuje možnost zavedení změn během procesu vývoje. Základní principy inkrementálního přístupu: [2]

Jsou prováděny série malých vodopádů, kde každý vodopád je prováděn pro malou část systému a je dokončen před pokračováním na další přírůstek, nebo

obecné požadavky jsou definovány dříve než se přikročí k evolučnímu vývoji pomocí malých vodopádů pro jednotlivé přírůstky systému, nebo

Prvotní koncept, analýza požadavků, design architektury a systémové jádro jsou definovány vodopádovým přístupem, následuje iterativní prototypový přístup, který vrcholí instalací konečného prototypu jako funkčního systému.



Prototypování

Vývoj pomocí vytváření prototypů je takový přístup k vývoji software, kde dochází k vývoji neúplných verzí software, tzv. prototypů. Základní principy prototypového přístupu[2]:

Není samostatným a kompletním přístupem metodiky vývoje, ale spíše přístup k jednotlivým částem větších tradičních metodik vývoje software (tj. přírůstková metoda, spirála, nebo RAD - Rapid application development).

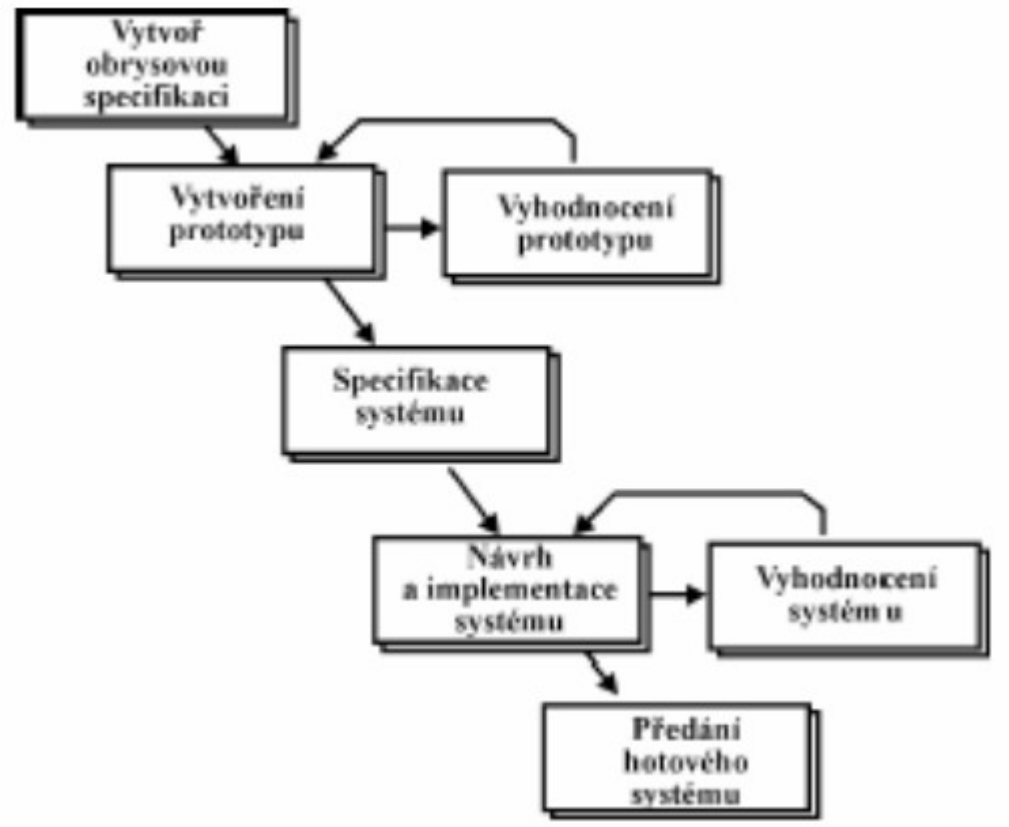
Snaha snížit nebezpečí projektových rizik rozdělením projektu na menší části a zjednodušit tak možnost změn v průběhu procesu vývoje.

Uživatel je zapojen v celém procesu vývoje, což zvyšuje pravděpodobnost přijetí konečné implementace uživatelem.

Malé ukázky systému jsou vyvíjeny iterativním procesem, dokud se prototyp nevyvine tak, že splňuje požadavky uživatele.

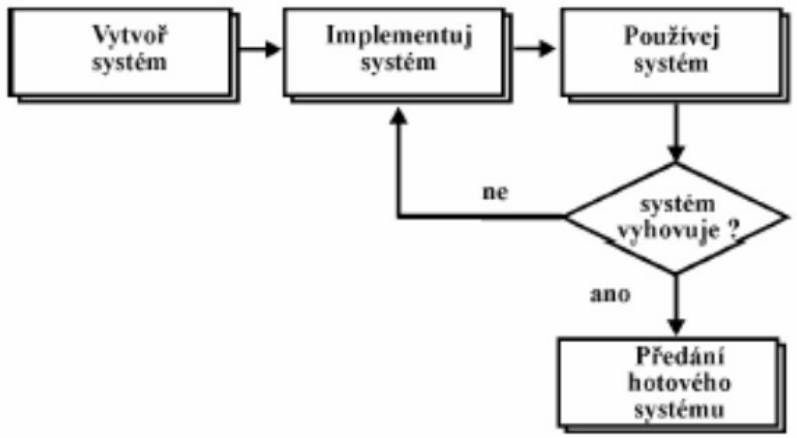
Většina prototypů je sice vyvíjena s tím, že budou vyřazeny, ale v některých případech je možné pokročit od prototypu k funkčnímu systému.

Aby se předešlo vývoji, který řeší jiný problém, než bylo zadáno, je třeba pochopit základní business problematiku.



Výzkumník

Pro bohaté firmy, které si mohou dovolit zlepšovat svůj produkt, implementovat něco nového.



Odlišnosti SW projektů:

Jistota produktu:

Je určená dvěma faktory:

1. Zda jsou uživatelské požadavky jasně specifikovány (zahrnuje funkcionalitu i kvalitu)
2. Jaká je zranitelnost těchto požadavků.

Jistota procesu

Faktory:

1. Možnost přesměrovat vývojový proces
2. Stupeň měřitelnosti procesu
3. Znalost efektu řídících akcí
4. Stupeň použití nových, neznámých nástrojů

Jistota zdrojů

Hlavním faktorem je dostupnost správných kvalifikovaných pracovníků.

4 základní řídící situace

	1.	2.	3.	4.
Jistota produktu	vysoká	vysoká	vysoká	nízká
Jistota procesu	vysoká	vysoká	nízká	nízká
Jistota zdrojů	vysoká	nízká	nízká	nízká

Realizační problém

- Jak pro dané požadavky dosáhneme cíle co nejefektivněji
- Lze použít lineární procesní model (vodopád), zpětné kroky slouží pro verifikační a validační aktivity
- Použití kalibrovaného cenového modelu
- **Stanovení ceny** je hlavním cílem manažera

Alokační problém

- Hlavním problémem je dostupnost lidských zdrojů
- Co největší standardizace procesu podpoří zaměnitelnost pracovníků
- Popisy pracovních postupů
- Lineární procesní model rozšířený o analýzu citlivosti (co se stane s cenou/dobou projektu, pokud přiřadíme 3 pracovníky kateg. A místo 4 pracovníků kategorie B)

Návrhový problém

- Požadavky, postup a potřebné zdroje
- Je potřeba navrhnout projekt, identifikovat milníky, potřebné dokumenty, zvážit profese, přiřadit odpovědnosti
- Pro „uřízení“ tohoto typu projektu je potřeba mít k dispozici nadbytečné kapacity rozpočtové i časové
- Nutnost měření postupu
- Nahrazení lineárního modelu inkrementálním
- Cenové modely založeny pouze na datech z předchozích projektů
- Nutnost citlivostní analýzy

Výzkumný problém

- Experti z různých oblastí pracují na základě vzájemné dohody na řešení přesně nespecifikovaného cíle
- Kritickým faktorem je shoda a flexibilita všech pracovníků projektu
- Management se musí zaměřit na mezilidské vztahy, nelze klást důraz na rozpočet
- Cílem je maximalizovat výstup s danými postředky
- Maximalizace se může zaměřit na kvalitu a/nebo funkcionalitu produktu
- Procesním modelem může být prototypovoání a řešení mnoha malých kroků
- Někdy se používají pilotní projekty

Plánovací a odhadovací nástroje

Work Breakdown Structure, textová nebo grafický diagram:

1. Úvodní studie

1.1 Zaměření situace

1.2 Interview se zákazníkem

1.3 Zpracování návrhu

1.4 Přednesení návrhu

1.5 Zpracování připomínek

2. Cenový odhad

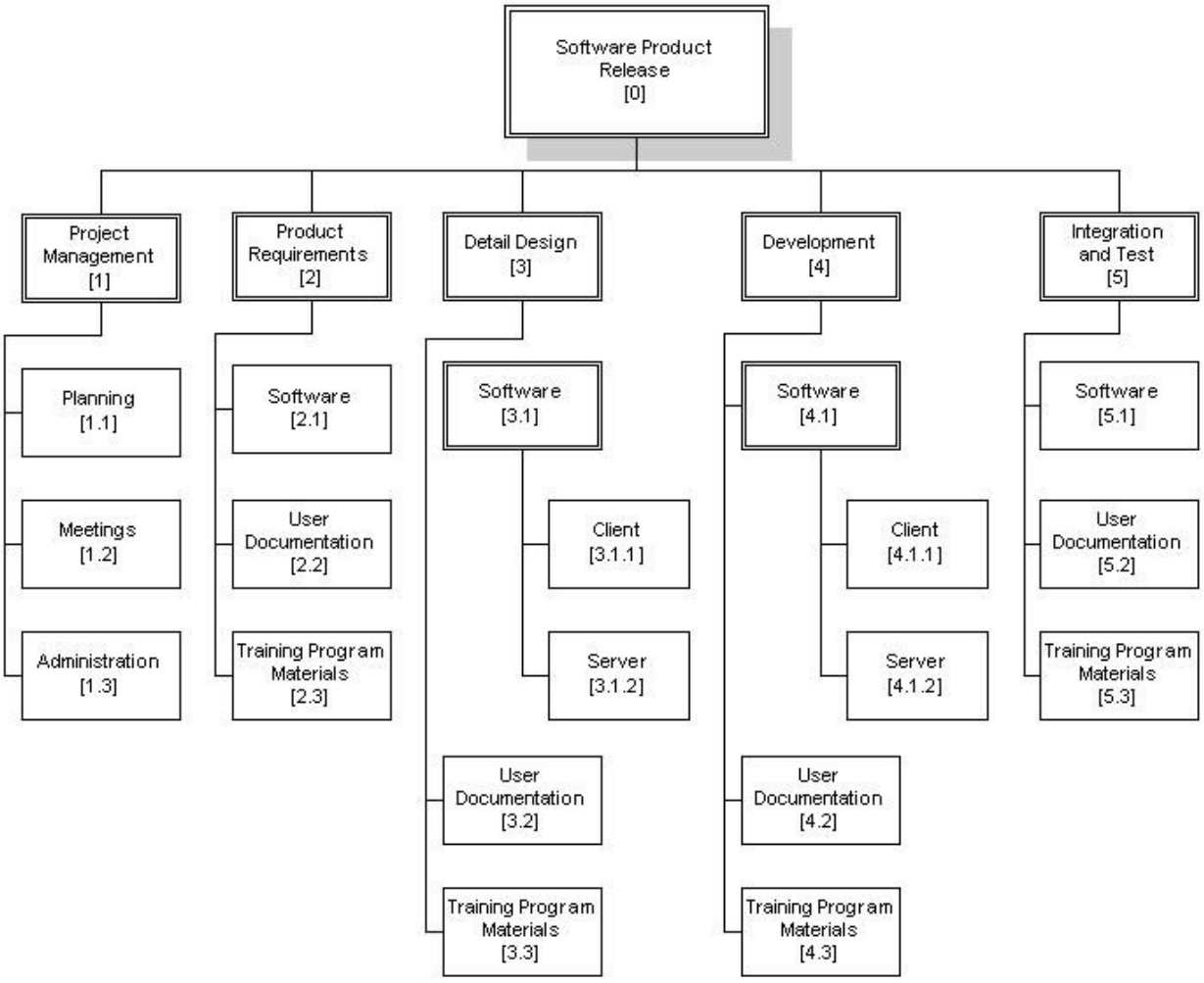
2.1 Případové studie a návrhy

2.2 Volba materiálu

3. Uzavření obchodní smlouvy

3.1 Příprava smlouvy

3.2 Podpis smlouvy



Sample Work Breakdown Structure organized by phase

Grafická podoba

Odhadování

Zdola-nahoru

- Větší množství práce, ale přesnější
- Často s expertním posudkem na úrovni úloh

Shora-dolů

- COCOMO, funkční body
- Používá se v úvodních fázích
- Spolu s analogií + expertním posudkem

Analogie

- Srovnání s předchozím projektem

Expertní posudek

- Pomocí členů týmu, kteří provedou práci

- Nejběžnější technika společně s analogií
- Nejlepší pokud je konzultováno u více expertů

Odhadování

- Parametrické /kompromisy mezi základními přístupy – LOC & FUNKČNÍ BODY)
- Funkční body (nezávislé na technologii použité pro vývoj systému)
- Odhad znovupoužití

Úvodní fáze projektu

Proč?

- Účel, zakládací listina

Co a jak?

- WBS a další plánovací dokumenty (plán vývoje SW, řízení rizik, Řízení konfigurací)

Odhadování

- Velikost (množství/složitost) a práce (trvání)
- Iterace

Plánování

- Začíná společně s prvními odhady
- Iterace

Primární cíle:

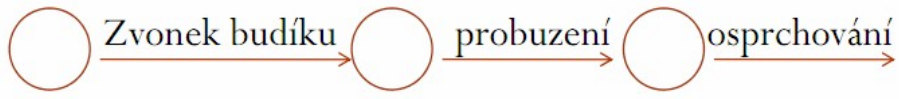
1 Nejkratší čas
2 Nejmenší cena
3 Nejnižší riziko

Druhotné cíle

4 Vyhodnocení alternativ plánu
5 Efektivní využití prostředků
6 Komunikace

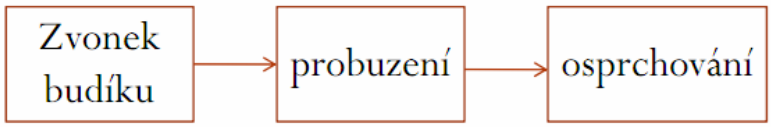
Síťové diagramy

AOA = Activity on Arrow

- Uzly zde reprezentují událost (start, cíl) a hrany činosti
- Využívají jej metody Critical Path Method (CMP) a Program Evaluation and Review Technique (PERT)
- 

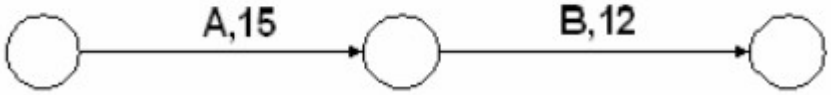
AON = Activity on Node

- Činnosti jsou v uzlech
- Informace o dané činnosti v uzlu

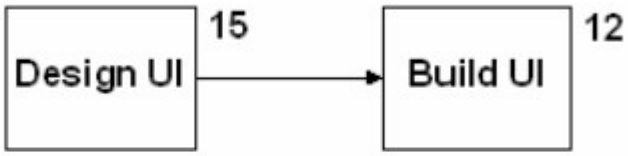
- Hrany udávají závislosti mezi činnostmi
- 

Activity on Arrow (AOA)

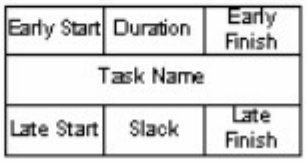
or



Activity on Node (AON)



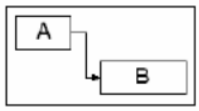
or



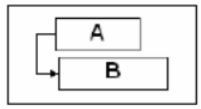
Každá úloha je označená pomocí identifikátoru (písmeno, kód) a trvání (standardní jednotky – např. dny)

- Jedna startovní a jedna koncová událost
- Čas zleva doprava

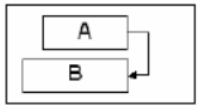
- **Doba trvání činnosti (Duration)** je doba, která je známá a je uvažováno s konkrétní časovou hodnotou (CMP), nebo ji lze odhadnout s určitou pravděpodobností (PERT).
- **Nejdříve možný začátek činnosti** (Early Start), je okamžik, v němž může činnost započít. U vstupní činnosti je nulový a u následujících vždy po uplynutí doby trvání všech činností, které do uzlu vstupují (nejdříve možný začátek + doba trvání činnosti).
- **Nejpozději přípustný začátek činnosti** (Late Start) je okamžik, v němž musí činnost začít, aby nedošlo k časovému skluzu. Je dán rozdílem nejpozději přípustného konce a doby trvání činnosti.
- **Nejdříve možný konec činnosti** (Early Finish) je okamžik, v němž může činnost nejdříve skončit. Je roven součtu nejdříve možného začátku a doby trvání dané činnosti. U počátečních činností je identický s dobou jejího trvání.
- **Nejpozději přípustný konec činnosti** (Late Finish) je okamžik, v němž musí činnost nejpozději skončit, aby nedošlo k ohrožení celkové doby trvání projektu. Je roven minimu z nejpozdějších přípustných začátků činností, které z uzlu vystupují. Jeho výpočet se odvíjí od vstupní činnosti.



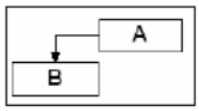
AON také rozlišuje tyto vztahy mezi činnostmi:



- Finish-to-Start (FS) zde platí, že entita B nemůže začít dokud entita A neskončí.
  - Příklad: A-postav dům, B- natři dům.
- Start-to-Start (SS) zde platí, že B nemůže začít, dokud A nezačne.
  - Příklad: A-lití betonu, B-rovnání betonu.



- Finish-to-finish (FF) zde platí, že B nemůže skončit, dokud A neskončí.
  - Příklad: A-přilij ocet, B-míchej majonézu.
- Start-to- finish (SF) zde platí, že B nemůže skončit dokud A nezačne.
  - Velmi málo používané.



Sestavení síťových diagramů probíhá:

- definováním jednotlivých činností procesu, jejich pořadí a vzájemné návaznosti (využití vývojového diagramu),
- stanovením časových charakteristik, jako je doba trvání jednotlivých činností a jejich časové návaznosti,
- sestavením příslušného diagramu, u něhož bude vypočtena celková doba trvání celého procesu, časové reservy a následně identifikovaná kritická cesta,
- sestavením časového diagramu, určením kontrolních milníků a návrh možných opatření.

3 podmínky tvorby síťového diagramu:

1 Kdo jsou PŘEDCHŮDCI? (které činnosti musí být dokončeny, aby mohla být zahájena další činnost?)

2 Kdo jsou NÁSLEDOVNÍCI? (které činnosti budou následovat)

3 Kdo jsou NEZÁVISLÍ (které činnosti mohou probíhat nezávisle na jiných činnostech)

4 typy závislostí mezi úlohami

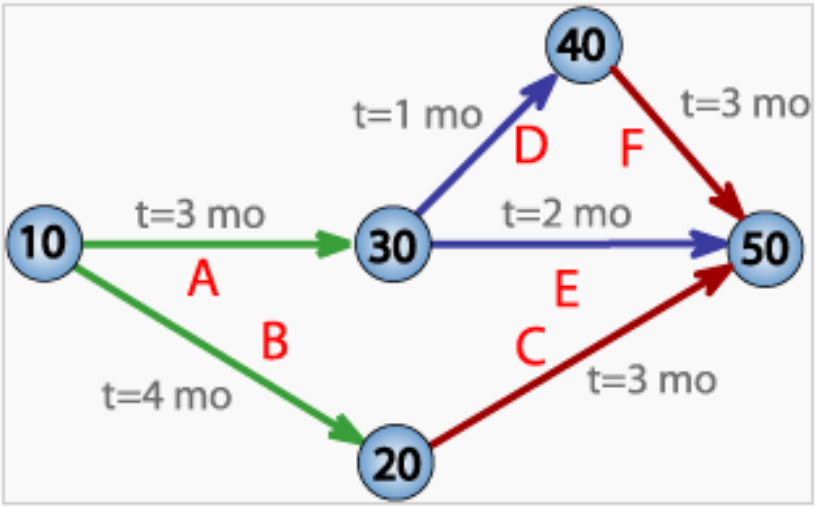
1 Povinné (postata práce diktuje uspořádání – kódování předchází testování, návrh UI předchází implementaci UI atp.)

2 Zvažované (určené manažerským týmem, řízené procesem; např. zvažované pořadí tvorby určitých modulů)

3 Vnější (mimo samotný projekt; zhotovení výrobku subdodavatelem, zrušení kontraktu, účastnící projektu, dodavatelé, konec roku)

4 Na zdrojích (dvě úlohy závisí na stejném zdroji; jedna databáze a více úloh)

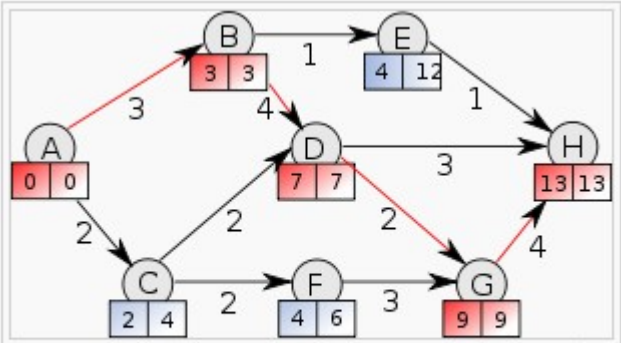
Kritická cesta



PERT síťový diagram pro projekt s pěti milníky (10 až 50) a šesti činnostmi (A až F). Projekt má dvě kritické cesty: B-C nebo A-D-F, minimální doba trvání tohoto projektu je tedy 7 měsíců (s použitím fast-trackingu). Činnost E je podkritická, tzn. může se zpozdit až o 2 měsíce, aniž by zpozdila projekt.

- Proces pro určení a optimalizaci kritické cesty.
- Úlohy mimo kritickou cestu mohou začít dříve nebo později, aniž by ovlivnily datum dokončení.

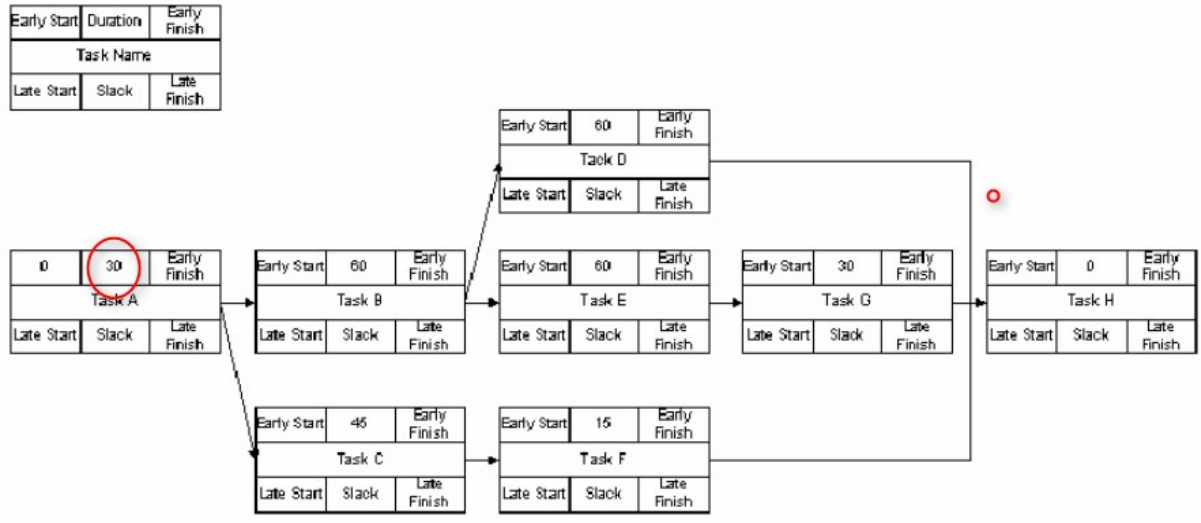
- Pro kritickou cestu pak platí, že nejdříve možný začátek je současně nejpozději možným začátkem a nejdříve možný konec je zároveň i nejpozději možným koncem. Opoždění kterékoliv činnosti na této cestě bude znamenat opoždění celého procesu. Předmětem a hlavním zaměřením takovéto analýzy jsou pak hlavně činnosti na kritické cestě, u nichž jsou hledány možnosti zkrácení.



Příklad grafu - červeně je vyznačena kritická cesta ABDGH.

První krok

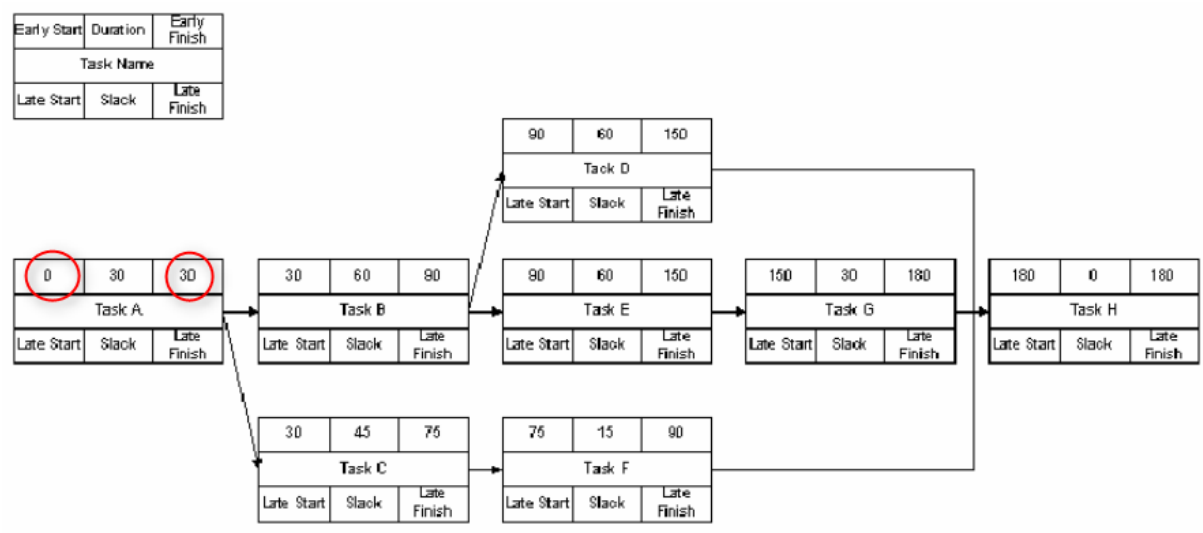
- Jsou vyplněny časové odhady úkolu místo „duration“



Druhý krok – dopředný průchod

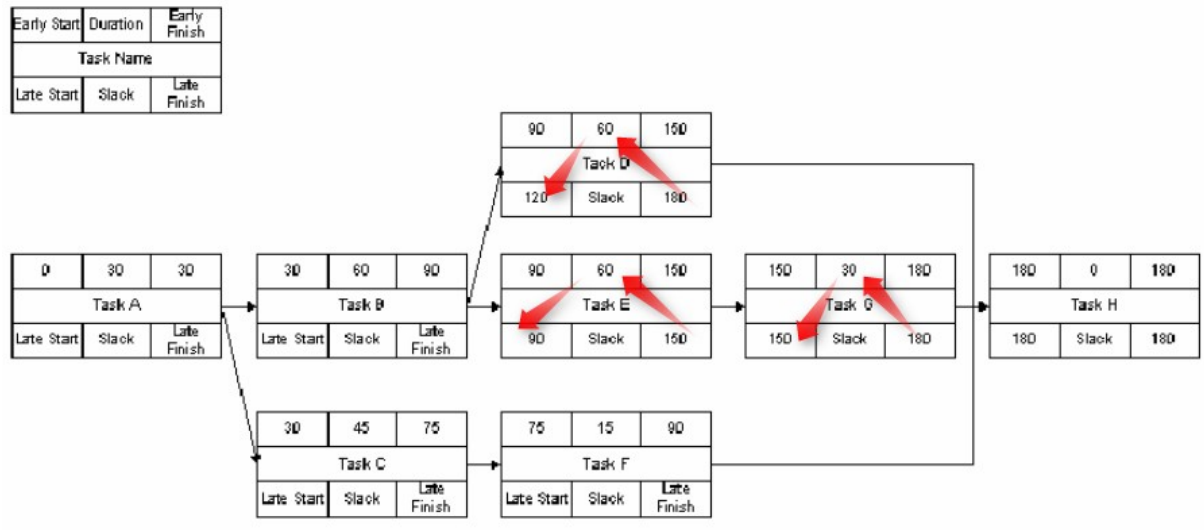
- Určuje časy brzkého začátku a brzkého konce (zleva doprava), přidávají se časy na každé cestě

- Pokud několik úloh konverguje do jednoho uzlu pak čas ES následující úlohy je roven největšímu z EF časů předchozích úloh



Třetí krok - Zpětný průchod

- Určuje čas nejpozdějšího konce (Last Finish – LF) a čas nejpozdějšího začátku (Last start – LS)
- Začíná se v koncovém uzlu – spodní páry čísel
- Odečte se trvání od času brzkého začátku v připojeném uzlu



A tak to pokračuje až se vyplní celá tabulka

PERT

Metoda PERT – je metoda, která se uplatňuje v situacích, ve kterých není možno stanovit deterministický odhad doby trvání jednotlivých činností. Tato metoda je pravděpodobnostním rozšířením metody CMP právě pro činnosti, které se neopakují.

Dobu trvání činnosti umožní odhadnout pouze s určitou pravděpodobností, proto jsou pro každou činnost zvažovány tři časové možnosti:

- optimistický odhad a – čas, který může být dosažen při zvlášť příznivých podmínkách a je přitom předpokládáno, že činnost neskončí v kratší době,
- modální odhad b – čas, který je za běžných podmínek dosahován nejčastěji
- pesimisticky odhad c – čas, který je očekáván při zvlášť nepříznivých, podmínkách a platí, že může být překročen jen a pouze v případě katastrofy.

Na základě určení těchto dob je možno vypočítat střední dobu trvání činnosti. Součtem středních dob na kritické cestě lze získat hodnotu celkové střední doby trvání pro celý projekt. Hodnotu střední doby trvání T (očekávaná doba) vypočítáme podle vzorce:

$$T = \frac{a + 4 \cdot b + c}{6}$$

Interval důvěry

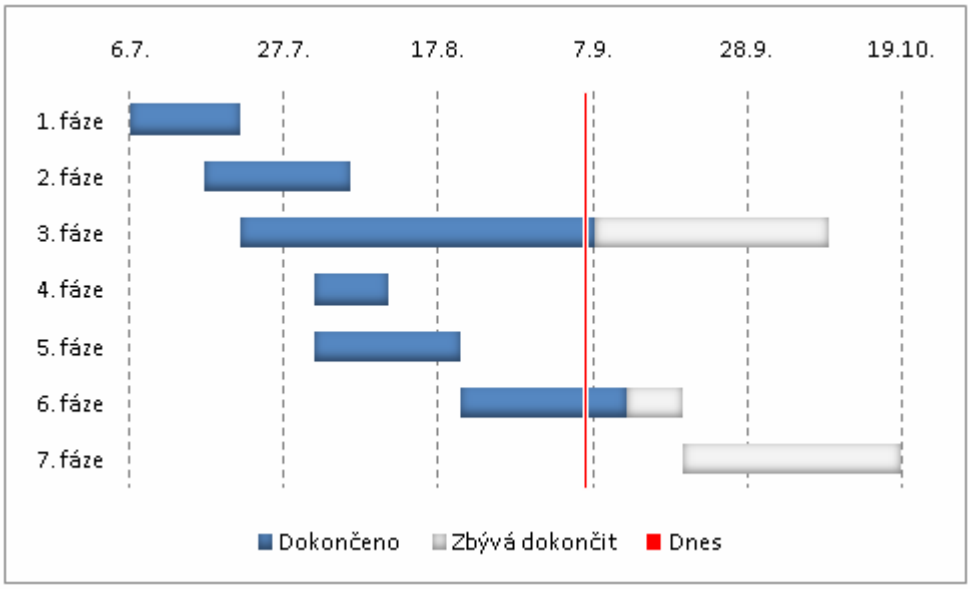
- Založeno na standardní odchylce očekávané doby
    - Normální rozložení
    - $$s = \frac{b - a}{6}$$
    - Pro celou kritickou cestu použijeme:
    - $$s_{cp} = \sqrt{s_1^2 + s_2^2 + \ldots + s_n^2}$$
    - Abychom měli 100% šanci na ukončení čas musí být zvýšen o 3 pravděpodobnostní odchylky daného odhadu:
    - Příklad:
        - Nejprve vypočítáme odhad podle PERT formule: 7 + 4x10 + 13/6 = 10
        - Pak vypočítám odchylku: (10-7)/6=0.5;
        - Zvýším očekávanou dobu o ty pravděpodobnostní odchylky tím trojitým znásobením: 10 + (3x0,5) = 11,5

Popis	Plán 1 (dny)	Plán 2 (dny)
m	10	10
a	9	9
b	12	20
čas PERT	10.16	11.5
Stand. odchylka	0.5	1.8

Ganttův diagram

Ganttův diagram se využívá při řízení projektů pro grafické znázornění naplánování posloupnosti činností v čase.

	A	B	C	D
1		Doba trvání	Datum zahájení	Datum dokončení
2	1. fáze	15	6.7.2009	=C2+B2
3	2. fáze	20	=C2+10	=C3+B3
4	3. fáze	80	=D2	=C4+B4
5	4. fáze	10	=C4+10	=C5+B5
6	5. fáze	20	=C4+10	=C6+B6
7	6. fáze	30	=D6	=C7+B7
8	7. fáze	30	=D7	=D4+10



Řízení rizik

Riziko = možnost ztráty nebo zranění (Webster)

E = P . Z

E = Expozice rizika
P = Pravděpodobnost nežádoucího výsledku
Z = ztráta při nežádoucím výsledku

Expozícia

	Probability	strata(tis CZK)	EXPO
1	10%	30	3
2	50%	150	75
3	1%	200-3 000	2 - 30
4	30%	30	9
5	20%	100	20
6	20%	30	6

1. E= 0,1*30
2. E=0,5*150
3. E=0,01*200 or 3000
4. E=0,3*30…..

Top 10 rizikové položky

1. Nedostatek pracovníků	6. Architektura, výkon, kvalita
2. Plány, rozpočty, proces	7. Změny v požadavcích
3. COTS, externí komponenty	8. Zděděný software
4. Neshody v požadavcích	9. Externě řešené úlohy
5. Neshody v uživatelském rozhraní	10. Přecenění možností informatiky

Otázky pro každou rizikovou položku

1. **Proč?**
    a. Význam rizikové položky, vztah k cílům projektu
2. **Co, Kdy?**
    a. Výstupy pro rozhodnutí rizik, milníky, sítě aktivit
3. **Kdo, Kde?**
    a. Zodpovědnosti, Organizace

4. **Jak?**
    a. Přístup (prototypy, přehledy, modely…)
5. **Kolik?**
    a. Zdroje (Rozpočet, plán, klíčoví pracovníci)

Příklad rizikové situace:

SW řídí vesmírný experiment

- PROB (pravděpodobnost kritické chyby) = 0.4
- Ztráta po výskytu kritické chyby = 20 mil.

Pokud se provede nezávislá V&V:

- PROB (kritická chyba zůstane) = 0.2
- Přidaná cena k SW projektu = 1 mil

Pokud se neprovede nezávislá V&V

- PROB (kritická chyba zůstane) = 0.2
- Přidaná cena k SW projektu = 0 mil

Risk Reduction Leverage

$$RRL = \frac{RE_{BEFORE} - RE_{AFTER}}{RISK\ REDUCTION\ COST}$$

· **Spacecraft Example**

	LONG DURATION TEST	FAILURE MODE TESTS
LOSS (UO) $PROB(UO)_B$ RE_B	$20M 0.2 $4M	$20M 0.2 $4M
$PROB(UO)_A$ RE_A	0.05 $1M	0.07 $1.4M
COST	$2M	$0.26M
RRL	$\frac{4-1}{2} = 1.5$	$\frac{4-1.4}{0.26} = 10$

Techniky řízení rizik:

- Identifikované riziko
    - Poškození auta a zranění pasažérů při autonehodě cestou do práce
- Vyhnutí se
    - Bydlet blízko práce a chodit pěšky, používat MHD
- Předpokládání
    - Jet do práce a doufat, že to dobře dopadne
- Řízení

    - Snížení maximální povolené rychlost, bezpečnostní pásy, zesílení oddělovacích panelů, cesta se zkušeným a bezpečným řidičem…

- Přenesení
    - Pojištění auta, záchranný systém, žaloba druhého řidiče
- Získání znalosti
    - Určení nejbezpečnějších aut pomocí nárazových testů, nalezení nejbezpečnější cesty do práce

COCOMO

- Odhadování ceny SW.
- V úvod. fázích se odhad může lišit až 4x na obě strany. Později zpřesňujeme.
- COCOMO vychází z pův. odhadu velikosti SW ve SLOC (KSLOC), počítá úsilí E(ffort) [člm] a dobu vývoje T(ime) [měs.]. Zdrojem empir. data z minulých projektů.

3 úrovně detailu:

Základní model (Basic COCOMO)

Využívá hrubý odhad E(KLOC) a T(KLOC). Je založen na odhadu velikosti softwaru KLOC. Odhad nákladů je však postaven na jednoduché klasifikaci projektu do tří typů a proto je odhadnutý rozpočet poměrně surový.

Střední model (Intermediate COCOMO)

Výpočty E(KLOC) a T(KLOC) jsou ovlivňovány různými faktory v závislosti na prostředí, ve kterém je projekt vyvíjen. Model bere v úvahu 15 atributů hodnocení driverů (cost drivers), které mají vliv na produktivitu, tudíž i na náklady.

Pokročilý model (Detailed COCOMO)

Výpočet je aplikován pro jednotlivé etapy životního cyklu. Model zohledňuje vlivy vývojové etapy, ve které se projekt v danou chvíli nachází. Stejně jako u středního modelu je i zde výpočet nákladů ovlivněn 15ti atributy hodnocení driverů.

3 vývojové módy:

Organický (Organic) typ

Tento typ zahrnuje spíše malé projekty, které jsou realizovány malými týmy. Postupy řešení jsou dobře známy, vychází ze zkušeností s řešením podobných projektů. Pro organický typ projektů jsou charakteristické mírné normy a malá omezení na specifikaci rozhraní. Velikost projektu nepřesáhne 50 tisíc řádků.

Bezprostřední (Semidetached) typ

Jedná se o typ mezi organickým a vázaným typem. Projekty mají velikost do 300 tisíc řádků. Tým má nepříliš velké zkušenosti s obdobnými projekty, řešená úloha je poměrně složitá, ale nehrozí žádná velká rizika. Jsou kladena zřetelná omezení pro uživatelská rozhraní.

Vázaný (Embedded) typ

Jedná se o projekty velkého rozsahu nad 300 tisíc řádků programu. Software musí pracovat za velmi přísných podmínek, vývoj projektu vyžaduje práci se složitými softwarovými a hardwarovými systémy. Obvykle jsou řešeny zcela nové problémy.

1 **organický (**do 50 KSLOC), - jednodušší dobře řešitelné projekty, zpravidla menšího rozsahu

1.1 Vědecké aplikace

1.2 Jednoduché obchodní modely a aplikace

1.3 Jednoduchá skladová aplikace

1.4 Jednoduchý systém pro řízení výroby

2 **bezprostřední** (do 300 KSLOC), - středně obtížné projekty

2.1 Transakční zpracování

2.2 Nový operační systém a překladač

2.3 Skladová aplikace střední složitosti

2.4 Systém pro řízení výroby střední složitosti

3 **vázaný (vše)** – rozsáhlé projekty s vysokými nároky na řízení

3.1 Složité transakční zpracování

3.2 Ambiciozní a složitý operační systém

3.3 RT aplikace

3.4 Složité povelové a řídící systémy

Úsilí a čas

Výpočet E(ffort)KSLOC a T(ime)(KSLOC):

$$E = a \,.(KSLOC)^b$$

$$T = c.E^d$$

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

Tabulka 3.1: Hodnoty parametrů pro základní model

Vývojový typ	*a*	*B*	*c*	*d*
Organický	2.4	1.05	2.5	0.38
Bezprostřední	3.0	1.12	2.5	0.35
Vázaný	3.6	1.20	2.5	0.32

Tabulka 3.2: Hodnoty parametrů pro střední a pokročilý model

Vývojový typ	*a*	*b*	*c*	*d*
Organický	3.2 F_C	1.05	2.5	0.38
Bezprostřední	3.0 F_C	1.12	2.5	0.35
Vázaný	2.8 F_C	1.20	2.5	0.32

Zmiňované zohlednění dalších faktorů projektu než pouze velikost kódu je ve středním a pokročilém modelu skryto v korekčním faktoru Fc. Ten je součinem 15ti dalších hodnot popisujících softwarové atributy produktu, hardwarové atributy, atributy vývojového týmu a

atributy projektu. Za normálních podmínek je každá z těchto patnácti hodnot rovna jedné. Pokud daný atribut má vliv na zvýšení ceny, vzrůstá i jeho hodnota. Jedná-li se o atribut, ke kterému není třeba příliš přihlížet, jeho hodnota naopak klesá. COCOMO nabízí tabulky (viz následující tabulka) obsahující konkrétní hodnoty atributů a definuje způsob jejich určení.

Intervaly hodnot parametrů:

- $a \in [2.4, 3.6]$ pro základní model
- $a \in [2.8\ F_c, 3.2\ F_c]$ pro střední a pokročilý model
- $b \in [1.05, 1.20]$
- $c = 2.5$ ve všech případech
- $d \in [0.32, 0.38]$

Tabulka A.1: Číselná hodnocení atributů produktu odhadu COCOMO

Atributy	*Hodnocení*					
	Velmi nízký	*Nízký*	*Normální*	*Velký*	*Velmi velký*	*Extrémně velký*
Atributy SW produktu						
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
HW atributy						
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
Atributy vývojového týmu						
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
Atributy metod vedení projektu						
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SCED	1.23	1.08	1.00	1.04	1.10	

Atributy SW produktu:

- RELY – míra požadavků na spolehlivost (0.75 – 1.40)
- DATA – míra rozsahu datové základny (0.94 – 1.16)
- CPLX – složitost produktu (0.70 – 1.65)

HW atributy:

- TIME – míra požadavků na dobu odezvy (1.00 – 1.66)
- STOR – míra využitelnosti paměti (1.00 – 1.56)
- VIRT – míra proměnlivosti OS počítače (0.87 – 1.30)
- TURN – míra rychlosti oběhu úlohy počítačem (0.87 – 1.15)

Atributy vývojového týmu:

- ACAP – míra schopnosti analytiků (1.46 – 0.71)
- AEXP – míra zkušenosti programátorů s podobnými aplikacemi (1.42 – 0.70)
- PCAP – míra kvality programátorů (1.29 – 0.82)
- VEXP – míra zkušenosti s počítačem (1.21 – 0.90)
- LEXP – míra zkušenosti s programovacím jazykem (1.14 – 0.95)

Atributy metod vedení projektu:

- MODP – míra použití moderních metod vývoje softwaru (1.24 – 0.82)
- TOOL – míra použití moderních prostředků vývoje softwaru (1.24 – 0.83)
- SCED – přesnost požadavků na dobu realizace (1.23 – 1.10)

Kroky při použití COCOMO

Výpočet nominálního úsilí En

1. První krok odhadu vychází z klasifikace typu softwarového projektu (organický, bezprostřední, vázaný typ) a úrovně modelu COCOMO (základní, střední, pokročilý model).

   Podle takto stanovených údajů se nastaví odpovídající hodnoty parametrů a, b a dosadí se do základního vzorce $E = a\,(KLOC)^b$ pro výpočet úsilí. Tím se určí hodnota nominálního úsilí En:

$$E_n = a\,KLOC^b$$

Výpočet korekčního faktoru FC

2. Stanoví se hodnocení vlivu faktorů reprezentovaných jednotlivými patnácti atributy. Určí se číselné hodnocení atributů produktu. Při základní úrovni modelu se tento krok vynechává,respektive korekční faktor FC je roven nominální hodnotě, tedy FC = 1 pro všechny atributy.

$$F_C = \prod_{i=1}^{15} Fi$$

Výpočet hodnoty úsilí E

3. Při základní úrovni modelu E = En. V případě střední a pokročilé úrovně modelu se při výpočtu zohledňují výše uvedené atributy produktu. Hodnota úsilí se vypočte ze vztahu

$$E = F_C \cdot E_n \quad [člověkoměsíce]$$

Výpočet doby vývoje T

4. Podle tabulky 3.1 a 3.2 se nastaví příslušné hodnoty parametrů c, d a dosadí se do vzorce pro výpočet doby vývoje projektu.

$$T = c\,E^d \quad [měsíc]$$

Hodnoty *a b c d,* jsou shodné pro střední a pokročilou úroveň modelu

- Pro střední úroveň se aplikuje výpočet na celý projekt
- Pro pokročilou úroveň se výpočet aplikuje pro jednotlivé etapy životního cyklu

COCOMO lze také použít pro odhad nákladů při modifikaci existujících aplikací:

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

- ESLOC = ekvivalentní počet SLOC
- ASLOC = odhadnutý počet modifikovaných SLOC
- DM = procento modifikace v návrhu

- CM = procento modifikace v kódu
- IM = integrační úsilí (procentu původní práce)

COCOMO II

Základem modelu COCOMO II jsou dva modely: Early Design model (EDM) a Post-Architecture model (PAM).

1. Early Design model lze využít v úvodních fázích projektu, kdy se teprve navrhuje architektura a není ještě tolik informací o samotném softwaru a jeho vývojovém procesu.

2. Post-Architecture model je detailnější a je určen pro použití ve fázi, kdy už je software připraven k samotnému vývoji. Oba dva tyto modely si jsou výpočtem velmi podobné a jsou oba navrhnuty pro odhad vývoje bez použití moderních RAD (rapid application development) a GUI (graphical user interface) nástroj ů pro rychlý vývoj softwaru. Mezi takové projekty patří například vývoj middleware řešení, vývoj nízkoúrovňového softwaru, integrace systémů atd. Pokud vývoj softwaru probíhá pomocí moderních RAD technik, existuje pro tento účel Application Point model (APM) jako rozšíření modelu COCOMO.

COCOMO II (1995) -3 různé modely:

1. Application Composition Model (ACM): pro projekty s použitím moderních nástrojů a GUI

2. **Early Design Model** (EDM): pro hrubé odhady v úvodních etapách, kdy se architektura vyvíjí

3. **Post Architecture Model:** pro odhady poté co byla specifikována architektura

Oba dva modely 2 a 3 používají stejný postup výpočtu, liší se pouze množstvím vstupních parametrů, které do vzorečku vstupují. Pro oba modely existují dva vzorce: jeden pro výpočet úsilí a druhý pro výpočet celkové doby trvání projektu. Výpočet úsilí probíhá dle následujícího vzorce:

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$
$$\text{kde } E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

Rovnice 4.6.1: COCOMO II - výpočet úsilí

$$\text{Úsilí} = (\text{multiplikátory okolí})[\text{velikost}]^{(\text{faktory procesu})} \quad (\sim E = a(KSLOC)^{b})$$
$$\text{Plán} = (\text{multiplikátor})[\text{Úsilí}]^{(\text{faktory procesu})} \quad (\sim T = c(E)^{d})$$

Velikost určena několika přístupy:

- KSLOC
- UFP (neupravené funkční body)
- EKSLOC (ekvivalentní velikost zdrojového kódu)

SF (Scale Factor): $SF = 1.01 + 0{,}01 \sum (w_i), \quad w_i - \text{hodnocení driverů exponentu}$

- drivery exponentu (hodnotíme 0.5) dle předch. výsledků, flexibility vývoje, rozhodnutí architektury/rizika, koheze týmu, vyspělosti procesu

- EM (Effort Multipliers, multiplikátory úsilí) – 7 pro ED, 17 pro PA

COCOMO II při modifikaci: $ESLOC = ASLOC \cdot (AA + SU + 0{,}4\ DM + 0{,}3\ CM + 0{,}3\ IM) / 100$

Funkční body

Je to normalizovaná metrika SW projektu (aplikační nikoliv technická oblast)
Typy funkčních bodů (dále jako FB):

1. FB vztažené k transakčním funkcím (EI – EO – EQ, viz dále)
2. FB vztažené k datovým funkcím (ILF, EIF)

Výpočet funkčních bodů

Nejdříve je třeba spočítat tzv. neupravené funkční body.

Ty jsou buď vztažené k transakčním funkcím, sem patří externí vstupy (EI - External Inputs), externí výstupy (EO -External Outputs) a externí dotazy (EQ - External Enquiry), nebo vztažené k datovým funkcím, což jsou vnitřní logické soubory (ILF - Internal Logical Files) a soubory vnějšího rozhraní (EIF - External Interface Files).

Všechny nalezené EI, EO, EQ, ILF a EIF aplikace se roztřídí do skupin podle typu a podle své složitosti. Počet prvků každé skupiny se vynásobí příslušnou váhou a následně se vše sečte. Výsledný součet dává počet neupravených funkčních bodů (NFP). Postup výpočtu funkčních bodů přesně stanoví koeficienty jednotlivých vah i postup, jak nalezené funkční body rozdělit do jednotlivých skupin.

Ropzis funkčních bodů:

Interní logické soubory – ILF

Každá velká logická skupina uživatelských dat nebo informací použitých pro řízení aplikace představuje jeden ILF. Zahrneme každý logický soubor, nebo v případě DB, každé logické seskupení dat z pohledu uživatele, které je vytvořeno, používáno, nebo udržováno aplikací.

Spíše než fyzické soubory započteme každé logické seskupení dat tak, jak je viděno z pohledu uživatele a jak je definováno při analýze požadavků nebo návrhu dat. Nezapočteme soubory, které nejsou přístupné uživateli prostřednictvím vnějšího výstupu nebo dotazu a které nejsou nezávisle udržovány.

Tučně zvýrazněné v následujícím textu je použití příkladu (vykradeno z jednoho projektu): **Skladové hospodářství, datové entity: Objednávka, Balík, zákazník, Aktuální stav skladu, Skladový prostor**

1. Logická entita nebo skupina entit z pohledu uživatele. (1 ILF) – **Odesílatel, Příjemce, Objednávka, Balík, Volné skladové prostory, Obsazené skladové prostory (5)**
2. Logický interní soubor generovaný nebo udržovaný aplikací. (1 ILF) – **Aktuální stav skladu (1)**
3. Uživatelem udržovaná tabulka(y) nebo soubor(y). (1 ILF) – **Přidání a úprava odesílatele příjemce, Přidání a odebrání skladových prostor (4)**
4. Datový soubor nebo soubor s řídící informací, který aplikace použije při sekvenčním zpracování a údržbě. (1 ILF)
5. Atributová entita udržovaná pouze prostřednictvím hlavní entity. (0 ILF)

6. Asociativní entity vytvořené průnikem nebo spojením obsahující pouze klíčový atribut (0 ILF)
7. Přechodný nebo třídicí soubor (dočasný soubor) (0 ILF)
8. Soubor vytvořený proto, že byla použita určitá technologie (např. indexový soubor) (0 ILF)
9. Soubor s předlohou (vzorem), který aplikace pouze čte. (0 ILF, 1 EIF) – **Přidání a úprava zákazníků, úprava objednávky (3 EIF)**

Soubory externího rozhraní

Započteme každou velkou logickou skupinu uživatelských dat nebo řídící informace používané aplikací. Tato informace musí být udržována jinou aplikací. Zahrňte každý logický soubor nebo logickou skupinu dat z pohledu uživatele. Započteme každou velkou logickou skupinu uživatelských dat nebo řídící informace, která je extrahována aplikací z jiné aplikace ve formě souboru externího rozhraní.

Extrakce nemá mít za následek změnu v některém z interních logických souborů. Pokud ano, pak započteme do EI místo do EIF.

1. Soubory nebo záznamy extrahované z jiné aplikace (použité pouze jako odkazy) (1 EIF)
2. Databáze čtená pomocí jiné aplikace (1 EIF)
3. Vnitřní logický soubor jiné aplikace použitý jako transakce (0 EIF, 1 EI)
4. Systém HELP, bezpečnostní soubor, chybový soubor čtený nebo odkazovaný aplikací, který pochází z jiné aplikace, která soubory udržuje (2 EIF)

Externí vstupy

Započteme každá unikátní uživatelská data nebo zadání uživatelských povelů, která vstoupí přes externí rozhraní do aplikace a přidá, mění, ruší nebo jinak pozmění data (např. přiřazení, přemístění, …) v interním logickém souboru. Započteme také řídící informaci, která vstoupí přes aplikační hranici a zajistí soulad s funkcí specifikovanou uživatelem. Externí vstup by měl být považován za unikátní, pokud logický návrh vyžaduje logiku zpracování odlišnou od ostatních externích vstupů.

1. Datová obrazovka s přidáním, změnou a rušením (3 EI) **Registrace a změna údajů zákazníka, vytvoření objednávky (3)**
2. Více obrazovek pohromadě zpracovaných jako jedna transakce (1 EI)
3. Dvě datové obrazovky s odlišným uspořádáním dat, ale se shodnou logikou zpracování (1 EI)
4. Dvě datové obrazovky se shodným formátem, ale s odlišnou logikou zpracování (2 EI)
5. Datová obrazovka s více unikátními funkcemi (1 EI za každou funkci)
6. Automatický vstup dat nebo transakcí z jiné aplikace (1 EI na každý typ transakce)
7. Vstup uživatelských povelů do aplikace (1 EI)
8. Vstupní formuláře (OCR) s jednou transakcí (1 EI)

9. Funkce úpravy dat, která následuje za dotazem (1 EI a 1 EQ)
10. Individuální výběry na obrazovce s menu (0 EI)
11. Oprava uživatelem udržované tabulky nebo souboru (1 EI) – **Změna údajů o zákazníkovi (1)**
12. Duplikát obrazovky, která již byla započtena jako vstup (0 EI)
13. Externí vstupy zavedené pouze kvůli technologii (0 EI)
14. Výběr položky ze seznamu (0 EI)

Externí výstupy

Započteme každá unikátní uživatelská data nebo řídící data, která opouští externí hranici měřeného systému. Externí výstup je považován za unikátní, pokud má odlišná data, nebo pokud vnější návrh (jiná aplikace) vyžaduje odlišnou logiku zpracování oproti jiným externím výstupům. Externí výstupy se často skládají z hlášení, výstupních souborů zasílaných jiné aplikaci nebo zpráv pro uživatele.

1. Výstup dat na obrazovku (1 EO)
2. Souhrnná zpráva - dávkové zpracování (1 EO)
3. Automatická data nebo transakce směrem k jiným aplikacím (1 EO) **Aktuální stav objednávek pro fakturaci (1)**
4. Chybové zprávy vrácené jako výsledek vstupní transakce (0 EO)
5. Záložní soubory (0 EO)
6. Výstup na obrazovku a na tiskárnu (2 EO) **Zobrazení a tisk objednávky (2)**
7. Výstupní soubory vytvořené z technických důvodů (0 EO)
8. Výstup sloupcového a zároveň koláčového grafu (2 EO) **Aktuální stav skladu (grafy) (2)**
9. Dotaz s vypočtenou informací (1 EO, 0 EQ)

Externí dotazy

Jako vnější dotaz započti každou unikátní vstupně/výstupní kombinaci, kde vstup je příčinou a generuje výstup. Vnější dotaz je považován za unikátní, pokud se od ostatních dotazů odlišuje typem výstupních datových elementů, nebo pokud vyžaduje odlišnou logiku zpracování v porovnání s ostatními externími dotazy.

1. On-line vstup a on-line výstup beze změny v datových souborech (1 EQ) **Aktuální stav skladu pro cracking a statistiku (2)**
2. Dotaz následovaný změnovým vstupem (1 EQ/1 EI)
3. Vstup a výstup na obrazovce s nápovědou (na dané úrovni)(1 EQ)

4. On-line vstup s bezprostředním tiskem dat beze změny dat (1 EQ) **Tisk stavu objednávky (cracking), tisk statistiky (2)**
5. Výběr ze seznamu nebo umístění s dynamickými daty (1 EQ)
6. Výběr ze seznamu nebo umístění se statickými daty (0 EQ)
7. Požadavek na zprávu obsahující neodvozená data (1 EQ)

Před výpočtem musíme EI, EO, EQ, ILF, EIF roztřídit do skupin podle vah:

Váhy	nízká	průměrná	vysoká	celkem
EI	___ x 3 +	___ x 4 +	___ x 6 =	_____
EO	___ x 4 +	___ x 5 +	___ x 7 =	_____
EQ	___ x 3 +	___ x 4 +	___ x 6 =	_____
ILF	___ x 7 +	___ x 10 +	___ x 15 =	_____
EIF	___ x 5 +	___ x 7 +	___ x 10 =	_____

Neupravené funkční body celkem _____

Matice složitosti vstupů (EI, EQ)

FTRs	1-4 DETs	5-15 DETs	16+DETs
0-1	nízká	nízká	průměrná
2-3	nízká	průměrná	vysoká
4+	průměrná	vysoká	vysoká

- FTR = File Types (User Data Groups) Referenced
- DET = Data Element Type (Attribute)
- RET = Record Element Type (User View)

Matice složitosti výstupů

FTRs	1-4 DETs	5-15 DETs	16+DETs
0-1	nízká	nízká	průměrná
2-3	nízká	průměrná	vysoká
4+	průměrná	vysoká	vysoká

- FTR = File Types (User Data Groups) Referenced
- DET = Data Element Type (Attribute)
- RET = Record Element Type (User View)

Matice složitosti souborů

RETs	1-19 DETs	20-50 DETs	51+ DETs
1	nízká	nízká	průměrná
2-4	nízká	průměrná	vysoká
5+	průměrná	vysoká	vysoká

- FTR = File Types (User Data Groups) Referenced
- DET = Data Element Type (Attribute)
- RET = Record Element Type (User View)

Obecné charakteristiky systému – stupnice hodnocení

Dále se zjišťuje 14 faktorů obecných charakteristik systému, jako je například míra požadované spolehlivosti zálohování a zotavení nebo míra požadovaných on-line vstupů dat. Každý faktor se hodnotí celým číslem do 0 do 5 podle stupně vlivu na aplikaci. Výsledný počet upravených funkčních bodů (FP) se získá podle následujícího vztahu.

0 = bez vlivu
1 = náhodný
2 = mírný
3 = průměrný
4 = významný
5 = podstatný

1. Vyžaduje systém spolehlivé zálohování a zotavení? 2. Jsou vyžadovány datové komunikace? 3. Existuje distribuované zpracování? 4. Je výkonnost kritická? 5. Poběží systém v stávajícím intenzivně využívaném operačním prostředí? 6. Systém požaduje on-line vstup dat? 7. Vyžaduje on-line vstup dat použití vstupní transakce přes více obrazovek nebo operací?	8. Jsou hlavní soubory opravovány on-line? 9. Jsou vstupy, výstupy, soubory a dotazy složité? 10. Je vnitřní zpracování složité? 11. Je kód navrhován s cílem znovupoužití? 12. Jsou konverze a instalace zahrnuty v návrhu? 13. Je systém navrhován pro násobné instalace u různých organizací? 14. Je aplikace navrhovaná tak, aby zajistila změny a snadné používání na straně uživatele?

Každou charakteristiku ohodnotíme [0..5]:

Výpočet

$$\text{Počet funkčních bodů} = [0.65 + (0.01 \times \textit{součet hodnocení charakteristik systému})] \times [\textit{počet nepřizpůsobených funkčních bodů}]$$

Postup výpočtu FP

1 Identifikujte a spočtěte ILF, EIF, EI , EO, EQ. Pro každou ILF a EIF identifikujte počet RET a počet DET. Pro každou EI, EO a EQ, identifikujte počet FTR a DET

2 S použitím matice složitosti spočtěte počet jednoduchých, průměrných a složitých položek EI, EO, EQ, ILF, EIF.

3 Spočtěte Počet neupravených funkčních bodů.

4 Určete hodnoty 14 charakteristik systému.

5 Sečtěte charakteristiky a určete Faktor technické složitosti systému.

6 Určete Počet upravených FP systému.

Příklad tabulky pro výpočet FP:

funkce	typ	váha	# souborů (FTR)	# datových elementů (DET)	# typů záznamů (RET)	složitost	skóre
vytvoření účtu	EI	1	1	4	×	nízká	3
úprava účtu	EI	1	1	4	×	nízká	3
zrušení účtu	EI	1	1	4	×	nízká	3
přihlášení	EI	1	1	2	×	nízká	3
odhlášení	EI	1	1	1	×	nízká	3
vytvoření rezervace	EI	1	7	5	×	vysoká	6
úprava rezervace	EI	1	2	5	×	průměrná	4
zrušení rezervace	EI	1	2	7	×	průměrná	4
vybrané rezervace pro další zpracování	EO	1	1	6	×	nízká	4
prohlídka linek	EQ	1	1	2	×	nízká	3
prohlídka jízd a jejich dostupností	EQ	2	2	8	×	průměrná	8
prohlídka rezervací	EQ	1	2	6	×	průměrná	4
uživatel	ILF	1	×	4	1	nízká	7
linka	ILF	1	×	4	2	nízká	7
jízda	ILF	1	×	4	1	nízká	7
stanice	ILF	1	×	4	2	nízká	7
Počet neupravených FP	×	×	×	×	×	×	76

Chyby SW

Výsledek projektu

1. **Úspěšný**: Projekt je dokončen včas, bez překročení rozpočtu, se všemi specifikovanými rysy a funkcemi.

2. **S výhradami**: Projekt je dokončen a funkční, ale překročil rozpočet, opožděný, méně rysů a funkcí, než bylo původně specifikováno.

3. **Neúspěšný**: Projekt je zastaven před dokončením, není implementován, nebo vyřazen po instalaci.

Prevence proti neúspěchu projektu

- ZAPOJENÍ vrcholového řízení a koncových uživatelů
- POUŽITÍ efektivního řízení projektu se spoluúčastí a zapojením vrcholového řízení na přezkoumáních
- POUŽITÍ efektivního řízení požadavků
- POUŽITÍ inkrementálního vývoje
- UČINĚNÍ "všech smysluplných kroků" při inženýrských aktivitách, t.j. dokumentace, měření, plánování, sledování, řízení kvality...

-> **Menší pravděpodobnost (totálního) neúspěchu projektu**

Spolehlivost SW – nemělo by být v testu

Porucha

Neschopnost systému nebo systémové komponenty provádět požadovanou funkci ve specifikovaných hranicích. Porucha může nastat, když se narazí na chybu, jejímž výsledkem je ztráta očekávané uživatelské služby.

Četnost poruch

1. Poměr počtu chyb dané kategorie nebo významu k časovému intervalu; např. poruchy za měsíc. (jiný název: intenzita poruch)
2. Poměr počtu poruch k dané jednotce měření; např. poruchy za jednotku času, poruchy pro daný počet transakcí, poruchy pro daný počet běhů programu.

Fault - chyba (defekt)

1. Chyba v kódu, která může být příčinou jednoho nebo více selhání.

2. Náhodná podmínka, která způsobuje, že funkční jednotka selhává při plnění požadované funkce. (synonymum: bug)

Error - chyba (omyl)

Nesprávná nebo chybějící akce uživatele, která zapříčiní chybu (defekt) v programu.

Četnost chyb na začátku systémových testů je v rozsahu 1 až 10 chyb/KSLOC, s průměrem 6 chyb/KSLOC.

KSLOC - počet předaných proveditelných řádek zdrojového kódu, bez znovupoužitého kódu, deklarací dat, komentářů apod. Očekávaný počet chyb odstraněných při opravě jednoho selhání: 0.955 chyb

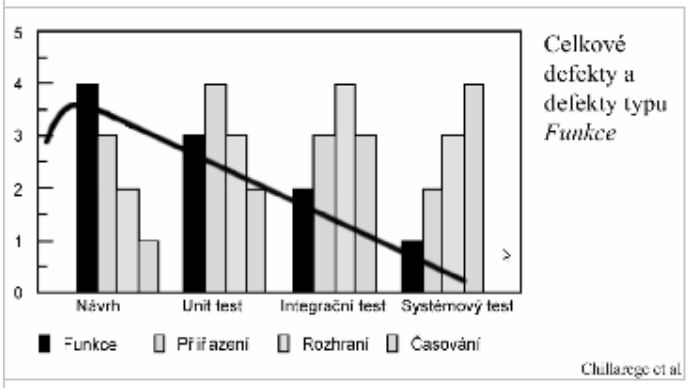
Některé druhy chyb:

- algoritmická chyba
- chyba syntaxe
- chyba výpočtu a přesnosti
- chyba dokumentace
- chyba stresu nebo přetížení
- chyba kapacity nebo meze
- časová nebo součinnostní chyba
- chyba propustnosti nebo výkonu
- chyba zotavení
- chyba HW a systémového SW
- chyba nedodržení standardů a procedur

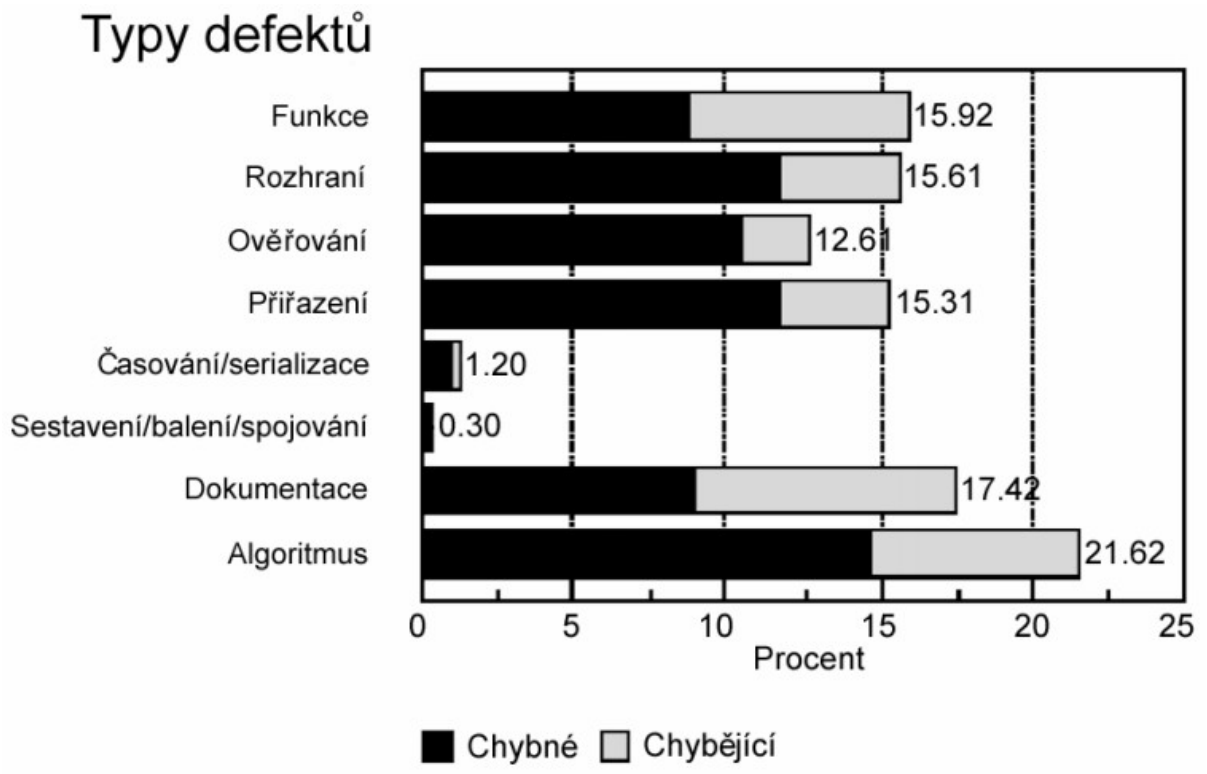
IBM – ortogonální klasifikace defektů – tohle asi v testu nebude

Typ defektu	Význam
• funkce • rozhraní • ověřování • přiřazení • časování/serializace • sestavení/balení/spojování • dokumentace • algoritmus	• význ. ovliv. schopnosti, rozhraní, glob. dat. strukt. • interakce s jinými komponentami • logika programu (neuspěla validace dat) • inicializace řídících bloků/dat. struktur • chce to lepší řízení sdílených a RT prostředků • „bordel“ ve správě knihoven, verzí, řízení změn • ovliv. publikace, údržbovou dokumentaci • probl. efektivity nebo správnosti (návrh se nemění)

Typ defektu	Vývojová etapa
• funkce • rozhraní • ověřování • přiřazení • časování/serializace • sestavení/balení/spojování • dokumentace • algoritmus	• návrh • návrh na nízké úrovni • návrh na nízké úrovni, kód • kód • návrh na nízké úrovni • knihovní nástroje • publikace • návrh na nízké úrovni



Celkové defekty a defekty typu *Funkce*

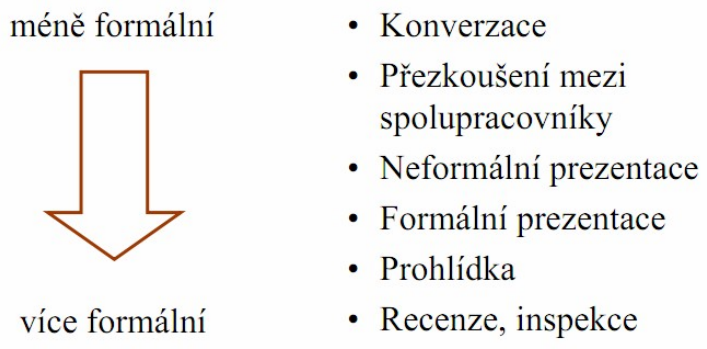


Obrázky jsou jen na ukázku, do testu to snad těžko narvou.

Inspekce – tohle asi taky v testu nebude

Recenze software z pohledu zajištění kvality
(SQA - Software Quality Assurance)

Efektivita přezkoušení



Efektivita přezkoušení (roste s formálností).

Různé review techniky

TYP METODY	TYPICKÉ CÍLE	TYPICKÉ VLASTNOSTI
Prohlídky	Minimální náročnost Školení vývojářů Krátká doba	Malá/žádná příprava Neformální proces Žádné měření Žádné FTR *(Formal Technical Review)*
Odborné recenze	Zjištění požadavků Rozlišení nejednoznačností Školení	Formální proces Představení autora Rozsáhlá diskuze
Inspekce	Účinné a efektivní zjištění a odstranění všech defektů	Formální proces Kontrolní seznam Měření Fáze přezkoušení

Inspekce&recenze prohlídky

	INSPEKCE, RECENZE	PROHLÍDKA
Požadavky	Výchozí požadavky	Detailní požadavky
Plány	Plány vývoje	
Vývoj	Návrh podrobného kódu	Návrh systému
Publikace		Pracovní & finální publikace
Testování		Implementace testu

Zkušební/inspekční tým: moderátor, zapisovatel, autor, recenzenti/inspektoři

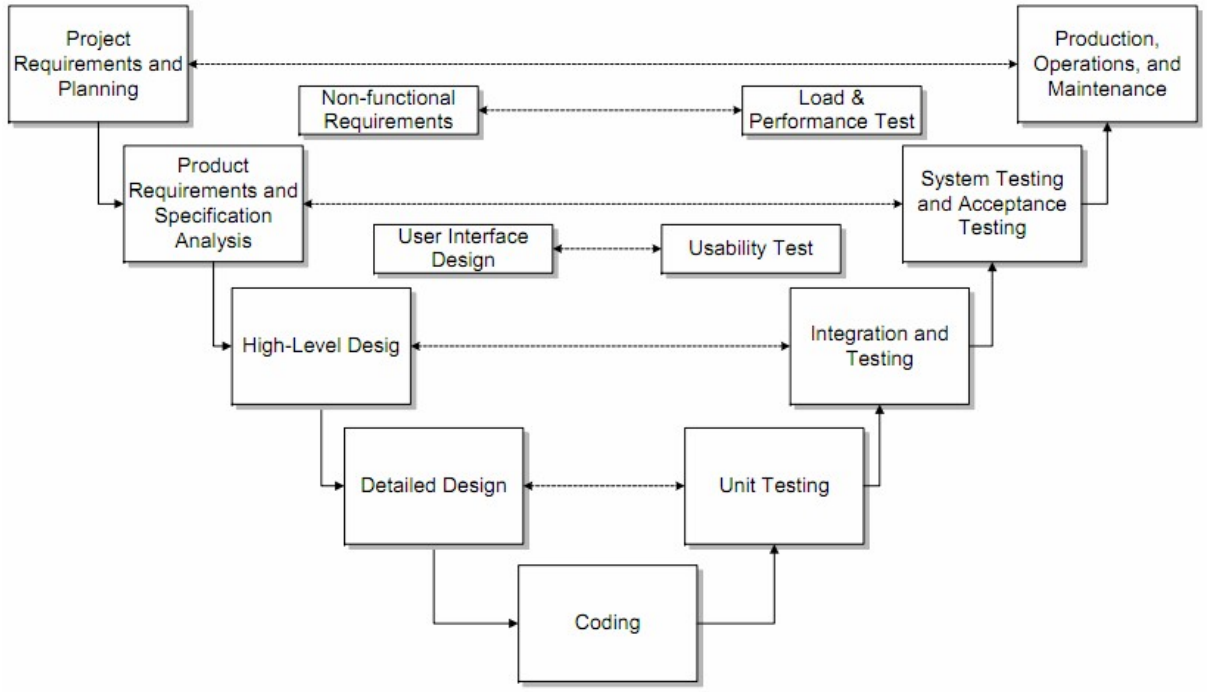
Dále k inspekci…

- **Příprava inspektora** – rozumí kontextu, projde si materiály, opatří poznámkami, připomínky jako otázky, nehodnotit styl, info v příp. nemožnosti se připravit.
- **Cíle**: odhalit chyby ve fci, logice, implementaci SW; ověřit zda zkoumaná položka splňuje požadavky; zda jsou splněny standardy; zajistit jednotný vývoj; zvýšení řiditelnosti projektů
- Vyhodnocení, přezkoušet produkt (ne tvůrce), klid, otázky, udržet agendu přezkoušení, uvádět nikoliv řešit problémy, tech. Správnost nikoliv styl, zaznamenat a ohlásit výsledky přezkoušení.
- Závažnost chyb, defektů: kritické (pád systému), vážné (odstranitelné), středně záváž. (část funkcionality), málo závaž. (nenaruš. fčnost).
- Používat formulář! (filtrace, jas. formulace prob., stats, pořadí důlež., ef. a správ. kontrol. seznamu.
- Ověř. produktů až jsou na to připraveny, termíny přísné (zvládnutelné),komentář autora zařazen pouze pokud je užitečný, kontrol. Seznamy a další materiály jsou užitečné, schůzka zaměř. na detekci problémů, autor se necítí ohrožen, úpravy prověřeny, každý se chce účast. znovu.

Testování SW produktů

- Testování je proces spuštění programu s cílem nalézt chyby.
- Dobrý testovací případ má vysokou pravděpodobnost nalezení dosud nenalezené chyby.
- Úspěšný test je takový, který odhalí dosud neodhalenou chybu
- Def: Test je úspěšný, pokud zjistí přítomnost jedné, či více chyb v programu.

Procesní V model (není potřeba umět)



Co testování ukazuje?

1 Testování nemůže ukázat nepřítomnost defektů, může pouze ukázat, že v softwaru jsou chyby.

2 Testování také ukazuje funkce a výkon.

3 A je také ukazatelem kvality software.

Verifikace&Validace

Každý inženýrský výrobek může být testován dvěma způsoby:

1 test proti specifikovaným funkcím = Validace “Dělat správné věci”

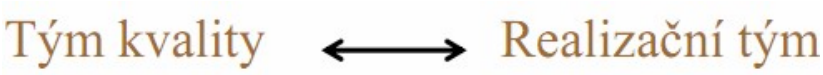
2 test proti vnitřní činnosti = Verifikace “Dělat věci správně”

Validace (dělá to to, co má; dává správné výsledky) vs. verifikace (dělá věci správným způsobem).

O testování v týmu…

- Testování je destruktivní činnost!

- Programátor není dobrým testerem vlastního výtvoru.
- Detailní znalost struktury programu usnadňuje hledání a opravu chyb.
- Je nutná spolupráce dvou nezávislých, organizačně samostatných týmů.



Provádět úplné testování není realizovatelné, je proto potřeba na to jít selektivně. (Např. program se 100 řádky, několik vnořených cyklů, každý proveden 20krát. Existuje přibližně 10^{14} možných cest, které mohou být provedeny. – Při rychlosti 1 test/ms by testování trvalo asi 3170 let…)

Selektivní testy

- I tehdy, kdy úplné testování není reálné (prakticky vždy!), testování „bílá skříňka“ by nemělo být vynecháno.
- Důležité logické cesty a cykly by měly být testovány.
- Selektivní testování validuje rozhraní a vytváří důvěru ve vnitřní činnost software

Dynamické testování

- Provedení počítačového programu s předem určenými vstupy.
- Porovnání dosažených výsledků s očekávanými výsledky.
- Testování je vlastně vzorkování, nemůže absolutně prokázat absenci defektů.
- Každý software má vši, testování nezaručí odvšivení.

Testovací případy

- Klíčové položky plánu testování.
- Mohou obsahovat skripty, data, kontrolní seznamy.
- Mohou mít vztah k *Matici pokrytí požadavků.*
    - Nástroj pro sledování

Černá a bílá skříňka

„černé skříňky“	FUNKCE	• tst. činnosti každé fce; letadlo letí? • ne jak to pracuje, ale co to dělá (V/V) • tst. případy založ. na specifikaci požadavků
„bílé skříňky“	VNITŘNÍ PRÁCE	• všechny motory pracují? • zohled. strukturu programu • provedené příkazy, cesty průchodu kódem • výpoč. cyklomat. složitosti, najít nezáv. cesty

Testování „černá skříňka“

- Funkční testování
- Program je “černá skříňka”
- Nezajímá nás, jak to pracuje, ale co to dělá.
- Zaměřeno na vstupy & výstupy
- Testovací případy založené na SRS (specifikacích požadavků)

Příklad:

Funkce

s = objednePresSMS (string telefon, string zeStanice, string doStanice)

Možné vstupy

```
prázdné ::- null, ""
vadný ::- (neexistující číslo), (špatný formát)
správný ::- (správné, korektní telefonní číslo)
_ ::- (cokoliv)
```

Test

1. s = objednejPresSMS(prázdné, _, _)
2. s = objednejPresSMS(vadné, _, _)
3. s = objednejPresSMS(správné, prázdné, _)
4. s = objednejPresSMS(správné, vadné, _)
5. s = objednejPresSMS(správné, _, prázdné)
6. s = objednejPresSMS(správné, _, vadné)
7. s = objednejPresSMS(správné, staniceA, staniceA)
8. s = objednejPresSMS(správné, staniceA, staniceB) //kde staniceA != staniceB
9. s = objednejPresSMS(správné, staniceA, staniceB) //ze stanice A do B neexistuje spojení

Další příklad, objednávka:

Funkce zajišťující kompletaci vyplnění všech adresních údajů. V případě, že je vše OK následuje výzva na zadání způsobu dopravy a platby.

f := vyplnAdresniUdaje (jmeno, prijmeni, ulice,mesto,psc,stat,telefon,mail, stavObjednavky)

Test č.	jmeno	prijmeni	ulice	mesto	psc	telefeon	mail	stavObjednavky
1	Václav	Klaus	Komenského	(null)	23245	823232321	Vasek212@gmail.com	Vyplňte město
2	Tomáš	Řepka	Einstainova	Praha	213221	679872932	Repka1@hmail.com	Zadejte způsob dopravy a platby
3	Radovan	Krejčíř	Seyšelská	Praha	232512	666276123	(null)	Vyplňte mail
4		Kostka	Brněnská	Praha	213221	762832762	Kostka21@gmail.com	Vyplňte jméno
5	Tomáš	Hanák	Jelcinova	Brno	(null)	823231123	Hanak27@gmail.com	Vyplňte psč
6	(null)	(null)	Kralupksá	Praha	321987	232123123	Grygar12@gmail.com	Vyplňte jméno, příjmení
7	David	Hollý	(null)	(null)	(null)	(null)	(null)	Vyplňte ulici,město,psč,telefón,mail
8	(null)	(null)	(null)	(null)	(null)	(null)	(null)	Vyplňte adresní údaje

Testování „bílá skříňka“

- Zohledňuje strukturu programu
- Pokrytí
    - provedené příkazy
    - cesty průchodu kódem
- vývojáři programují testy, test suites (kolekce testů), tst. postupně během vývoje, příp. po dokončení indiv. Modulů

Integrace & testování – nemělo by být v testu

- postupně propojuje funkcionalitu, QA tým spolupracuje s vývoj. týmem
- Integrační postupy:
- shora dolů: jádro (kostra), minim. skořápka, protézy; v souladu s prototypováním
    - složité objekty/moduly nelze zaměň. za protézu, výsledky na vyšších úrovních ne vždy viditelné
- zdola nahoru: individuální moduly sestav. zdola, kombin. do subsystémů, subsys. do celku; nadřazené objekty – „drivers“
    - náklady na konstrukci „drivers“ obvykle větší než u protéz, až v závěru prototyp použitelný k předvedení

Integrační testy: vývojový a/nebo QA tým. # pracovníků a rozpočet na vrcholu! „Jde do tuhého.“ Tlak, termíny, neočekávané bugy, motivace, konflikty se zákazníkem...

Měření SW – připravit si 4 metriky k testu (na proces, zdroj a produkt)

Klasifikace metrik

Produkty

- Explicitní výsledky činností vývoje software
- Dodávky, dokumentace, produkty

Procesy

- Činnosti související s výrobou software

Zdroje

- Vstupy do činnosti v oblasti vývoje softwaru
- Hardware, znalosti, lidé

Produkt vs proces

Procesní metriky:

- Podrobné informace o procesním paradigmatu, softwarové inženýrství úkoly, pracovní produkt, nebo milníky
- Vede k dlouhodobě k zlepšování procesů

Produktové metriky:

- Hodnotí stav projektu
- Sledování potenciálních rizik
- Odhaluje problémové oblasti
- Nastavení pracovního postupu nebo úkolů
- Vyhodnocuje týmovou schopnost k řízení kvality

Typy metrik

- Přímé metriky (interní atributy)
    - náklady, úsilí, LOC, rychlost, paměť
- Nepřímé metriky (vnější atributy)
    - Funkčnost, kvalita, komplexnost, efektivnost, spolehlivost, udržovatelnost

Metriky orientované na velikost

- velikost SW, SLOC, KLOC, úsilí (v člm), Errors/KLOC, Defects/KLOC, Cena/SLOC, strany dokumentace/KLOC; LOC závislé na použitém jazyce a programátorovi

Metriky orientované na funkce

- analýza FP, nepřímé měření, empirické vztahy založ. na přímých měřeních
- *FP = total count * [0.65 + 0.01 * Sum(Fi)], kde Fi (i z [1..14]) jsou hodnoty obecných charakteristik (viz 14bodů u FP)*

Metriky složitosti (komplexity)

LOC – fce komplexity; závislé na jazyce a programátorovi

Halstead's metrics (měření entropie):

Maurice Howard Halstead představil v roce 1977 další způsob, jak vyčíslit složitost programu. Jedná se o celou skupinu několika fomulí, a proto se v některých zdrojích tato metrika označuje jako Halstead's Software Science.
Halstead předpokládá, že každý program lze přepsat na seznam tokenů a každý z nich lze klasifikovat buď jako operátor, nebo jako operand.
n1 – poč. růz. operátorů; n2 – poč. růz. operandů

N1 – celk. poč. operátorů; N2 – celk. poč. operandů

Délka: $N=N_1+ N_2$

Slovník: $n=n_1 + n_2$

Odhadovaná délka: $\tilde{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2$

Příklad:

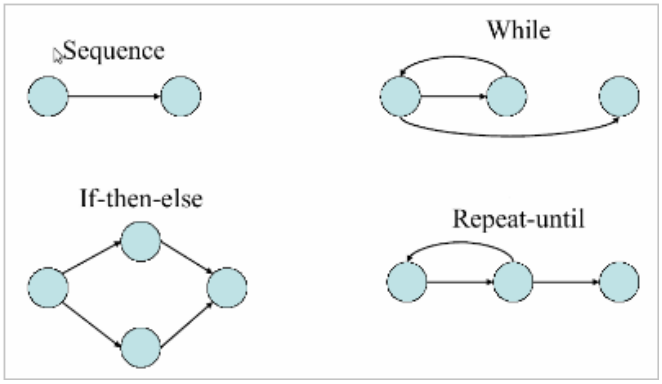
```
if (k < 2)
{
if (k > 3)
x = x*k;
}
```

- Distinct operators: if ( ) { } > < = * ;
- Distinct operands: k 2 3 x
- $n_1 = 10$
- $n_2 = 4$
- $N_1 = 13$
- $N_2 = 7$

McCabeho metriky komplexity:

Cyklomatická složitost byla prezentována v roce 1976 matematikem Thomasem J. McCabem, podle něhož bývá často nazývána. Metrika ve své podstatě udává počet všech cest, kterými lze program projít. K tomu, aby bylo možné cyklomatickou složitost defnovat, je potřeba

nejdříve zavést pojem Control Flow Graph.



- množina různých nezávislých cest
- V(G) = E – N + 2
    - E … počet hran
    - N … počet uzlů
- V(G) = P + 1
    - P … počet uzlů s podmínkou
- obtížné testování pro V(G) větší než 10

Výpočet V(G)

- V(G) = 9 – 7 + 2 = 4
- V(G) = 3 + 1 = 4
- Basic set
    - 1,7
    - 1, 2, 6, 1, 7
    - 1, 2, 3, 4, 5, 2, 6, 1, 7
    - 1, 2, 3, 5, 2, 6, 1, 7

Význam:

- V (G) je počet (uzavřených) regionů / oblasti rovinného grafu
- Počet regionů se zvyšuje s počtem rozhodovacích cest a smyček.
- kvantitativní míra testování obtížnosti a údaj o maximální spolehlivost
- Experimentální data ukazují, hodnota V (G), by neměla být větší než 10. Testování je velmi obtížné nad touto hodnotou.

McClureho komplexita:

- Komplexita = C + V

- C … poč. porovnání v modulu
- V … poč. řídících proměnných v modulu
- Podobné jako McCabe ale s ohledem na kontrolu proměnných

Metriky obecného návrhu

- strukturální, datová, systémová komplexita
- strukturální komplexita modulu i (provázanost na okolí, s kým si modul posílá zprávy)
  - $S(i) = f_{out}^2$, Fan out je poč. modulů přímo vyvolávaných modulem

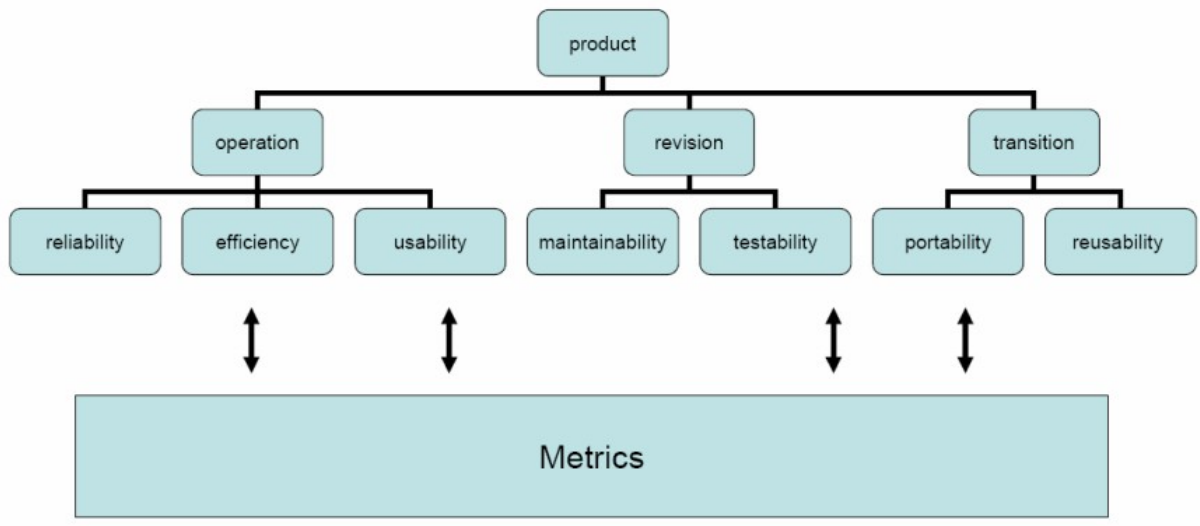
Metriky návrhu

- Datová komplexita
  - $D(i) = v(i)/[f_{out}(i)+1]$, *v(i)*… počet vstupů a výstupů modulu
- Systémová komplexita
  - $C(i) = S(i) + D(i)$

Metriky na úrovni komponent

- Koheze (provázanost uvnitř komponent)
- Párování (provázanost vůči okolí):
  - Data and control flow
    - d_i - input data parameters
    - c_i - input control parameters
    - d_o - output data parameters
    - c_o - output control parameters
  - Global
    - g_d - global variables for data
    - g_c - global variables for control
  - Environmental
    - w - fan in number of modules called
    - r - fan out number modules that call module
- Složitost toku řízení

Metriky kvality SW



FURPS

Functionality – Usability – Reliability – Performance - Supportability

- Funkce - funkce systému
- Usability – aesthesis (?), dokumentace
- Spolehlivost - frekvence selhání bezpečnost
- Výkon - rychlost, propustnost
- Schopnost podpory - udržovatelnost

měříme:

- korektnost (defekty/KSLOC, defekt/nedostatek shody s požadavky,chyby/hodinu práce)
- udržovatelnost (průměrná doby změny, žádosti o změnu na novou verzicena opravy...)
- integritu (chybová tolerance, bezpečnost&ohrožení,zotavení z vlastní chyby)
- použitelnost (čas školení, potřeba stupně zkušeností, zvýšení produktivity, dotazník...)

Objektově orientované metriky

- Chidamber & Kemerer '94 TSE 20 (6)
    - Metriky speciálně určené k řešení objektově orientovaného software
    - Třídou orientované metriky
    - Přímé metriky
- Velikost třídy:
    - Celkový počet operací
    - Počet atributů

    - Může být údaj o příliš vysoké odpovědnosti za třídu.

- NOO (Numper of Operations Overriden) – počet překrytých metod
    - Velký počet pro NOO naznačuje možné problémy s designem.
    - Nedostatečná abstrakce v hierarchii dědičnosti.

- NOA (Numper of Operations Added) – čím níže třída v hierarchii, poč. přid. operací by měl klesat

- Index specializace:
    - $SI = [NOO * L] / M_{total}$
        - L … úroveň třídy v hierarchii
        - M_{total} … celkový počet metod
        - vyšší hodnoty: třída v hierarchii neodpovídá abstrakci
    - Pak jsou tam další metriky jako method inheritance factor, coupling factor a tak. Na test budou stačit jen tři Vámi vybrané. Není potřeba se tedy nějak tohle drtit

Používání metrik

- zvolím problém → formuluji problém → sbírám data → analyzuji data → interpretuji data → zpětná vazba.

Příklady metrik na projektech:

Metriky projektu 1

1. FO - Počet typů použitých v deklaracích, formálních parametrech, návratových typech, deklaracích vyhazování vyjímek a lokálních proměnných. Jednoduché a rodičovské typy se nepočítají. FO využíváme k měření složitosti a "čistoty" návrhu.
2. NOAM - Počítá se množství metod přidané třídou. Zděděné a překrité metody nejsou započítány. Vysoká hodnota téte metriky indikuje velké odlišnosti od rodičovské třídy. V takovém případě je vhodné zvážit jestli by daná třída neměla být rozdělena do několika menších tříd. NOAM využíváme k ohodnocení znovupoužitelnosti kódu.
3. Měření produktivity zaměstnanců v závislosti na: pracovní době (flexibilní, pevná), pracovním prostředí (v kanceláři, doma). Výpočet: počet funkčních bodů za jednotku času.Dané metriky používáme pro zefektivnění produktivity a dodržování termínů.

Metriky projektu 2

1. Počet tříd/projekt (NOC per project)

   Víme, jakým stylem naši vývojáři programují. Protože máme zkušenosti z předchozích projektů, známe průměrnou velikost třídy pro určitý typ projektu a tedy nám počet tříd na projekt dokáže říct hodně o velikosti projektu a množství potřebného effortu pro projekt.

2. Průměrná cyklomatická složitost třídy (average CC per project)

Logická složitost kódu = počet větvení. Umožňuje odhadovat kolik procent z kódu je rozhodovací logika a kolik zbytek (přelévání dat atd.). Kód s rozhodovací logikou je mnohem náchylnější na vložení chyb, proto nám tato metrika poskytuje informaci o složitosti projektu.

3. Podíl privátních členů/třída (PPrivM per class)

   Tato metrika vypovídá o komplexnosti procedur, které jsou schované za veřejná rozhraní, a tím taky o effortu, který je potřeba pro implementaci těchto procedur. Rozhraní je většinou jasné už při analýze, velikost implementace rozhoduje o velikosti kódu.

Kvalita SW produktů – v testu jako CMM 2 a 3. Úrovně

Klasický pohled na kvalitu SW

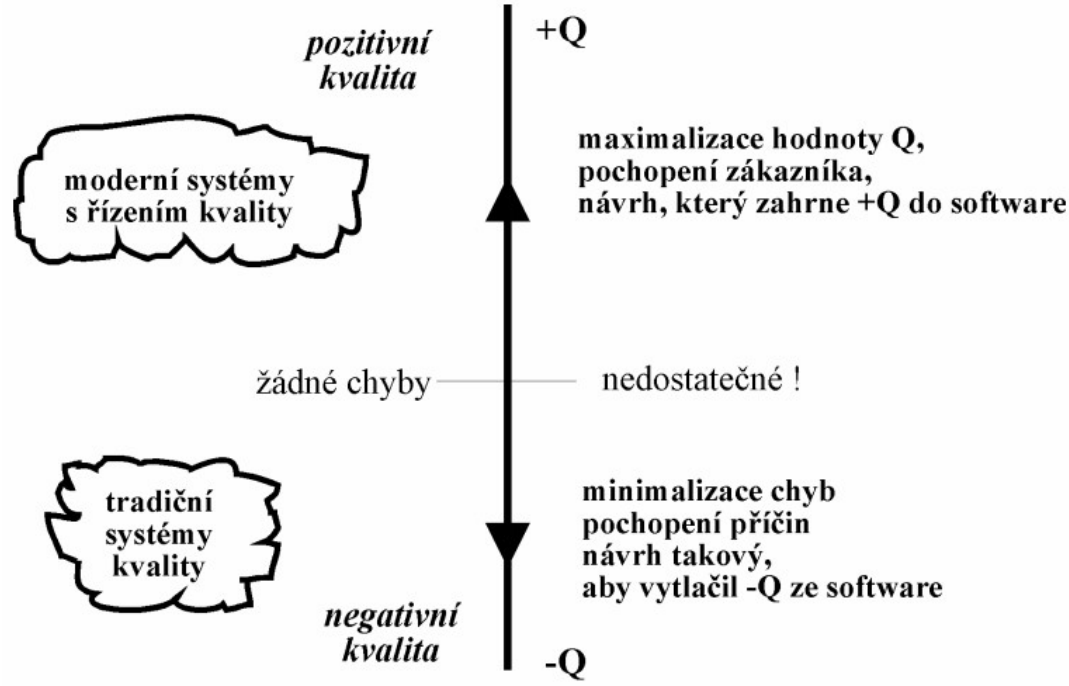
Každý program dělá něco správně; nemusí však dělat to, co chceme, aby dělal.

Kvalita: Dodržení explicitně stanovených funkčních a výkonových požadavků, dodržení explicitně dokumentovaných vývojových standardů a implicitních charakteristik, které jsou očekávány u profesionálně vyrobeného software.

Aspekty kvality:

- odchylky od požadavků na software
- nedodržení standardů
- odchylky od běžných zvyklostí (implicitních požadavků)

Nový pohled – spojité chápání kvality



Kvalita - IEEE Std. 610.12-1990

- Stupeň, do jaké míry systém, komponenta nebo proces splňuje specifikované požadavky.
- Stupeň, do jaké míry systém, komponenta nebo proces splňuje zákazníkovy nebo uživatelovy potřeby nebo jeho očekávání.

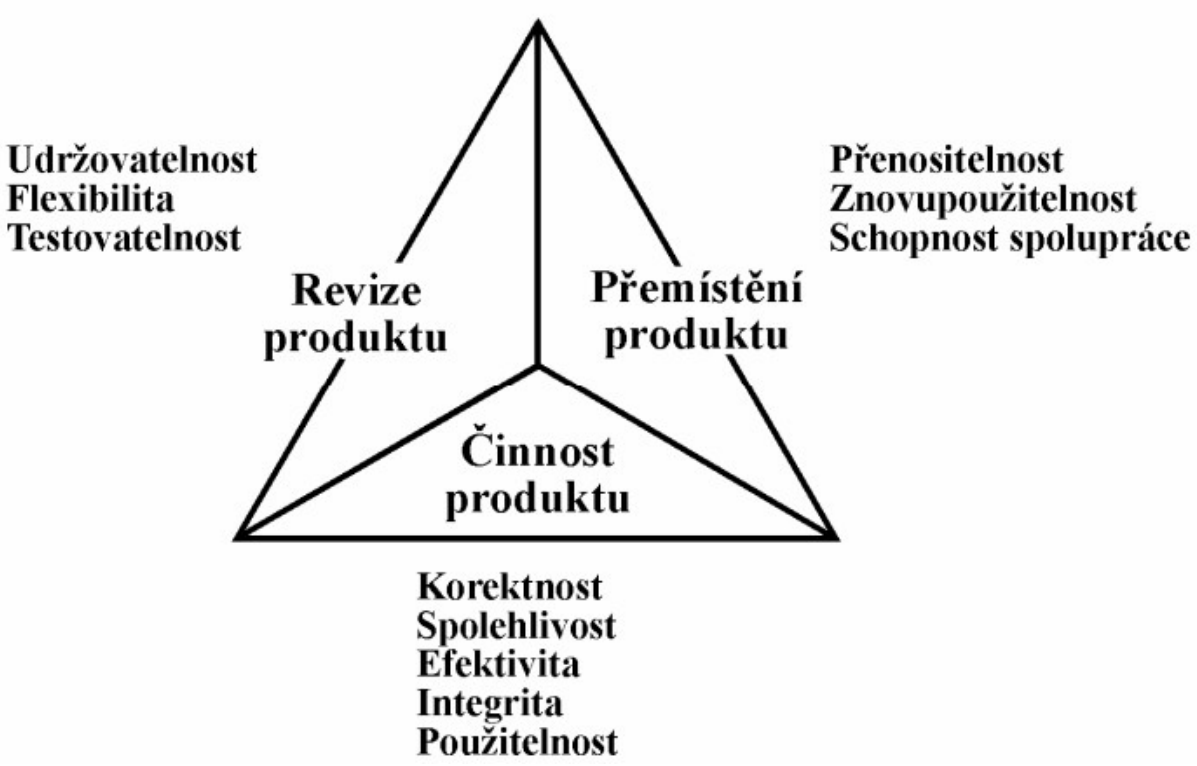
Faktory kvality SW

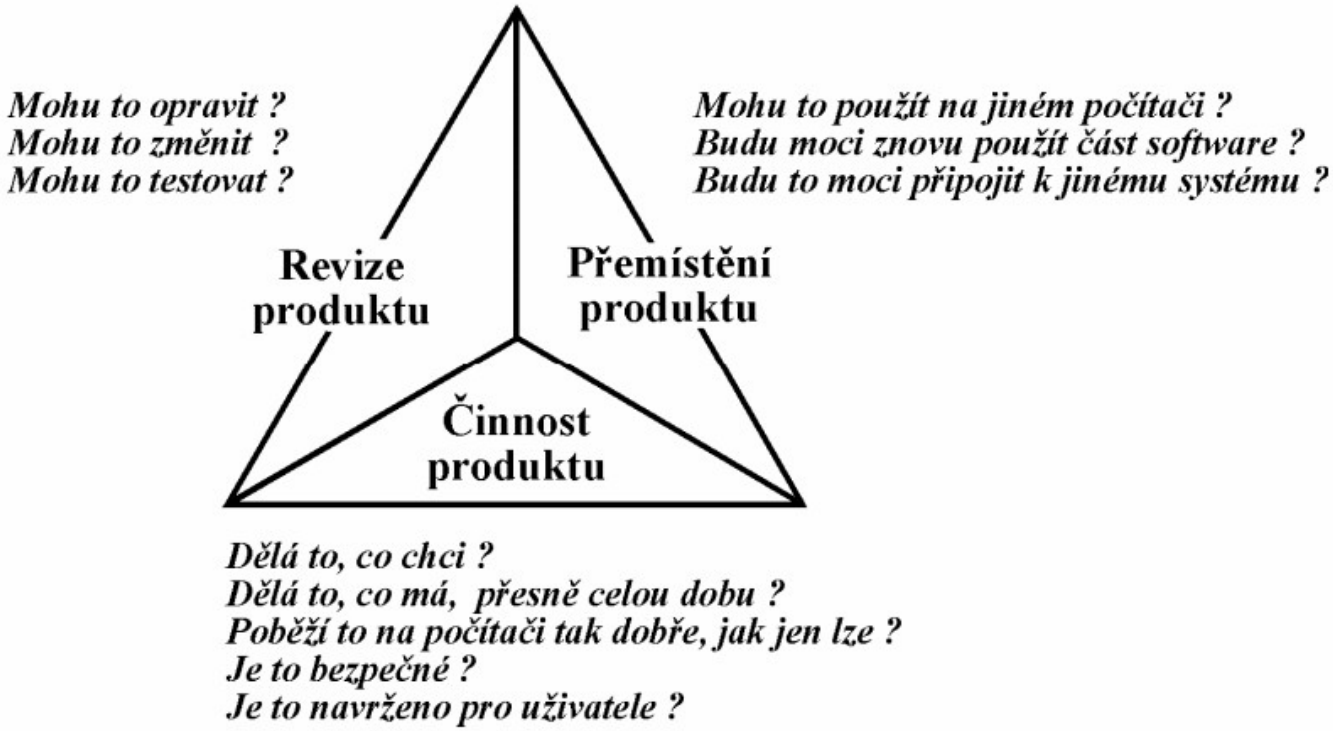
- Přímo měřitelné faktory
    - #chyb/KLOC/čas
- Pouze nepřímo měřitelné faktory
    - použitelnost, udržovatelnost
- Kategorie faktorů kvality:
    - operační charakteristiky
    - schopnost akceptovat změny
    - adaptibilita na nové prostředí

Faktory kvality - McCall et al. (1977)

- **Korektnost**: Rozsah toho, jak program splňuje specifikaci splňuje uživatelovy záměry.
- **Spolehlivost**: V jakém rozsahu lze očekávat, že program bude plnit zamýšlené funkce s požadovanou přesností.
- **Efektivita**: Množství výpočetních prostředků a kódu, které program potřebuje na splnění svých funkcí.
- **Integrita**: V jakém rozsahu mohou být program nebo data používána neoprávněnými osobami.
- **Použitelnost**: Úsilí vyžadované na učení, operování, přípravu vstupu a interpretaci výstupu programu.
- **Udržovatelnost**: Úsilí vyžadované na vyhledání a opravu chyby v programu.
- **Flexibilita**: Úsilí vyžadované na modifikaci provozovaného programu.
- **Testovatelnost**: Úsilí potřebné na testování programu tak, abychom se ujistili, že plní zamýšlené funkce.
- **Přenositelnost**: Úsilí potřebné na přemístění programu na jiný HW/SW.
- **Znovupoužitelnost**: Rozsah, v jakém lze program nebo jeho části znovu použít v jiné aplikaci (funkce a balení produktu).

- **Schopnost spolupráce:** Úsilí, které je nutné vynaložit pro připojení daného systému k jinému.





Globální hodnocení kvality výroby

- Vyspělost organizace: model CMM
- Systémy kvality: norma ISO 9001
- Ocenění kvality: cena MBNQA

CMM - Capability Maturity Model

Úroveň 1: Výchozí

- Chaotický proces, nepředvídatelná cena, plán a kvalita.

Úroveň 2: Opakovatelný

- Intuitivní; cena a kvalita jsou vysoce proměnlivé, plán je pod vědomou kontrolou, neformální metody a procedury.
- Klíčové prvky :
    - řízené požadavky
    - plánování softwarového projektu
    - řízené subkontrakty na software
    - zajištění kvality software
    - řízení softwarových konfigurací
    - také SEI model (Software Engineering Institute, Carnegie-Mellon Univ.), revize 1993

Úroveň 3: Definovaný

- Orientován na kvalitu; spolehlivé ceny a plány, zlepšující se, ale dosud nepředvídatelný přínos (výkon) systému kvality.
- Klíčové prvky:
    - zlepšování organizačního procesu
    - definice organizačního procesu
    - školicí program
    - řízení integrovaného software
    - aplikace inženýrských metod u softwarového produktu
    - koordinace mezi pracovními skupinami
    - detailní prověrky a oponentury

Úroveň 4: Řízený

- Kvantitativní; promyšlená statisticky řízená kvalita produktu.
- Klíčové prvky:
    - měření a kvantitativní řízení procesu výroby
    - řízení kvality

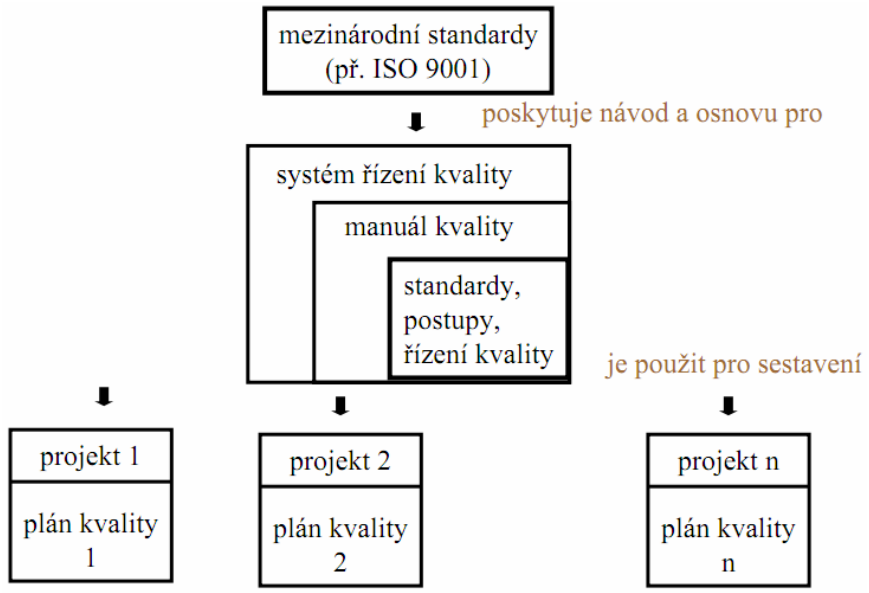
Úroveň 5: Optimalizující

- Kvantitativní základ pro kontinuální investice směřující k automatizaci a zlepšení výrobního procesu.
- Klíčové prvky:
    - prevence chyb
    - inovace technologie
    - řízené změny výrobních procesů

Příklady a lidské objasnění CMM

0 - neexistující řízení: Procesy a jejich řízení je zcela chaotické
1 - Počáteční (Initial): Procesy jsou realizovány adhoc
2 - Opakované (Repeatable): Dodržuje se určitá kázeň nezbytná pro provádění základních opakovaných procesů
3 - Definovaná (Defined): Procesy organizace jsou zdokumentovány
4 - Řízená (Managed): Procesy jsou řízeny a provádí se měření jejich výkonnosti pomocí KPI
5 - Optimalizovaná (optimized): Procesy jsou trvale zlepšovány, existuje inovační cyklus na procesech a řízení

Systémy kvality v řízení výroby



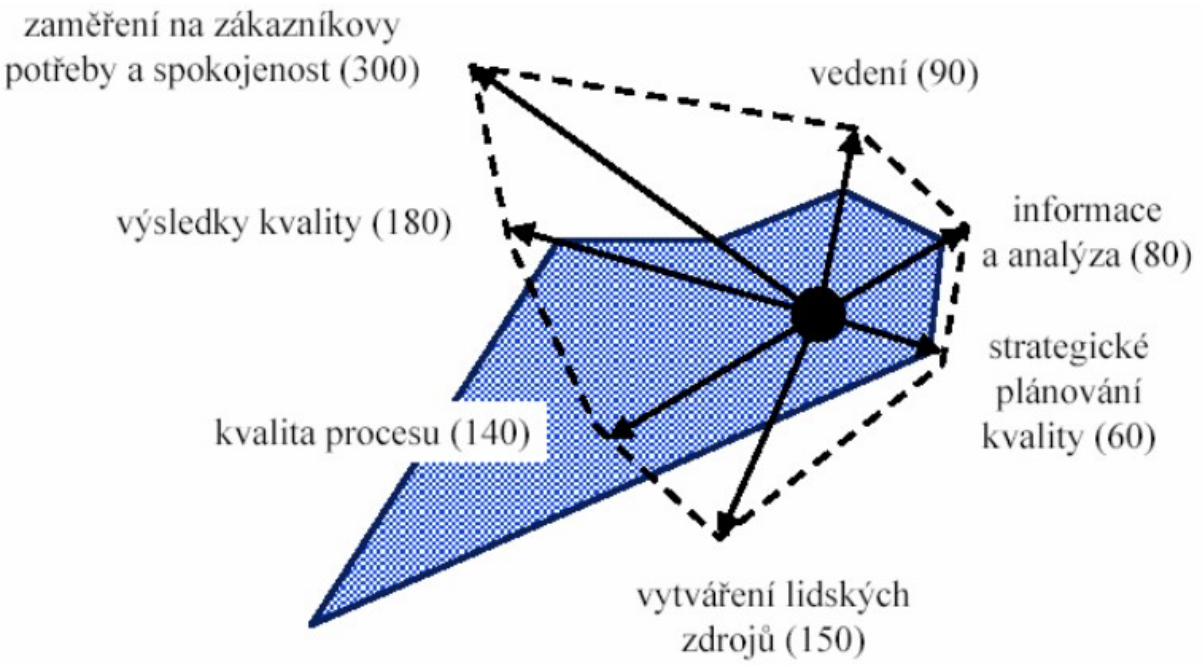
Principy systémů SQA

- Definovaná a dokumentovaná politika kvality a manažerský podíl
- Definice zodpovědností, autorit a vztahů mezi všemi osobami, které svojí prací mohou ovlivnit kvalitu
- Dokumentované postupy a instrukce pro kvalitu
- Efektivní implementace dokumentovaného systému kvality na všech úrovních organizace
- Záznam všech aktivit SQA

ISO 9001 - Systémy kvality

5. Zodpovědnost vedení
6. Systém kvality
7. Přehled zakázek
8. Řízení návrhu
9. Řízení dokumentace
10. Nakupování
11. Zakoupené produkty
12. Identifikace a sledování produktu
13. Řízení procesu
14. Inspekce a testování
15. Inspekční, měřicí a testovací vybavení
16. Stav inspekce a testování
17. Zvládnutí nevyhovujícího produktu
18. Opravné akce
19. Manipulace, skladování, balení a doručení
20. Záznamy o kvalitě
21. Vnitřní prověrky kvality
22. Školení
23. Služby

Vztah mezi MBNQA a ISO 9001



Jak na SQA?

- Formulace hypotézy
- Pečlivý výběr vhodných metrik
- Sběr dat
- Interpretace dat
- Iniciace akcí vedoucích ke zdokonalení
- Iterace s vyhodnocením vlivu přijatých opatření, formulace dalších hypotéz

Příklad jednoduchého sběru dat

Evidovaná činnost:

	plán	skutečnost	rozdíl	důvod
práce				
začátek				
konec				
trvání				

Příklad CMM

Příklad 1

CMM úroveň 2: Opakovatelný

Projekt splňuje požadavky druhé úrovně:

- řízené požadavky
  - Požadavky na software jsou součástí smlouvy. Vychází z konzultací se zákazníkem a zaměstnanci.
- plánování softwarového projektu
  - Činnosti týmu jsou detailně popsány v plánu projektu. Na jeho dodržování a změny dohlíží management.
- řízené subkontrakty na software
  - S projektem souvisí i použití stávajícího software, nákup hardware, instalace a testování.
  - V týmu jsou vyčleněni pracovníci pro tyto činnosti, které jsou součástí plánu.
- zajištění kvality software
  - Software je důkladně testován v každé fázi inkrementálního vývoje i na straně zákazníka.
- řízení softwarových konfigurací

Příklad 2

CMM úroveň 3: Definovaný

Pro dosažení třetí úrovně by bylo za potřebí naplnit tyto požadavky:

- zlepšování organizačního procesu
  - optimalizace procesů uvnitř firmy za pomoci metrik
- definice organizačního procesu
  - definice hierarchického postavení zaměstnanců
  - plán profesního růstu
- školící program
  - školení jsou zajištěna v rámci projektu
  - plán pravidelných školení pro zaměstnance
- řízení integrovaného software
  - plánované aktualizace podpůrného softwaru
- aplikace inženýrských metod u softwarového produktu
  - použití analytických a návrhových metod při vývoji
- koordinace mezi pracovními skupinami
  - zčásti koordinuje samotný plán činností
  - pomáhá firemní informační systém
- detailní prověrky a oponentury
  - inspekce, analýza metrik práce zaměstnanců

SW fyzika

Práce a délka programu

- N … délka programu (SLOC)
- T … spotřeba práce (člm, MM)
- P … produktivita P = N / T
- D … doba realizace programu
- S … prům. počet řešitelů

Práce na vytvoření programu

Odhad práce a regresní analýza:

$$\log T \simeq a + b \log N, \quad a, b - \text{vhodné konstanty}$$
$$T \simeq c\, N^b, \quad b \geq 9/8$$

Pracnost roste exponenciálně v závislosti na rozsahu programu. S rostoucí délkou klesá i produktivita programátorů.

Putnamova rovnice – byla na testu ☹

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

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

Důsledky Putnamovy rovnice

Pro řešení programu zvažujeme 2 projekty:

Projekt A: N_A, T_A, D_A Projekt B: N_B, T_B, D_B

$D_A < D_B$; např. $D_B = 1.2\ D_A$ => (empirie) $N_A > N_B$

Programy psané ve spěchu jsou delší.

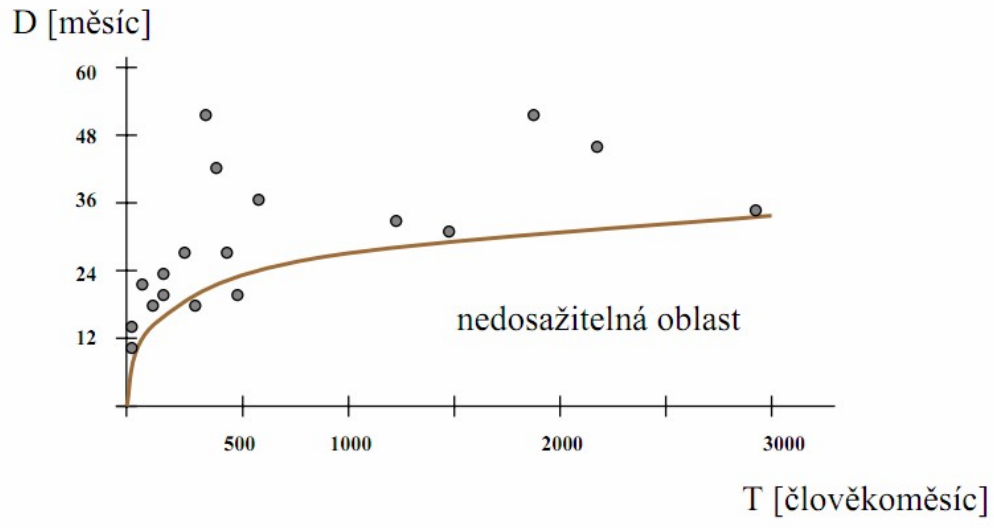
tedy $N_A / N_B > 1$

$$N_A / N_B \cong (T_A / T_B)^{1/3} . (D_A / D_B)^{4/3} > 1$$

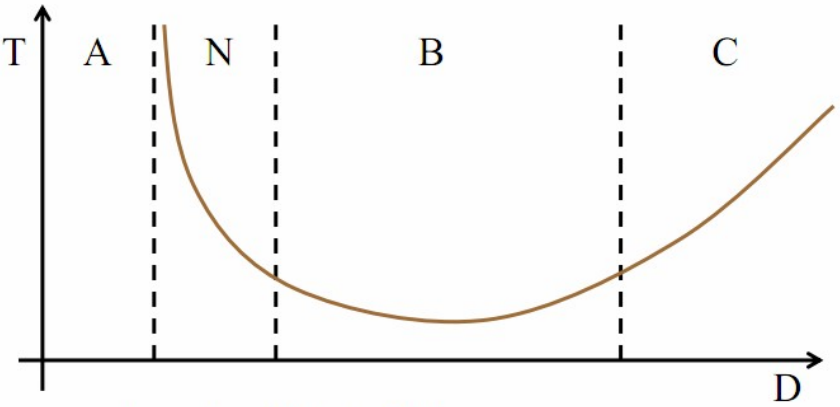
odtud $T_A > T_B . (D_A / D_B)^4$

Pro náš případ $T_A > T_B . 2{,}07$.

Při zkrácení termínu na 83% je pracnost je dvojnásobná.



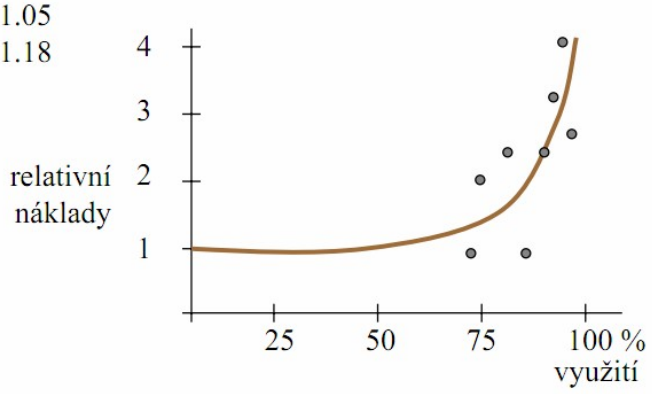
Pracnost a doba řešení



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

Pracnost a využití HW

stupeň využití	T	D
50%	1.00	1.00
60%	1.08	1.00
70%	1.21	1.00
80%	1.4	1.05
90%	2.25	1.18

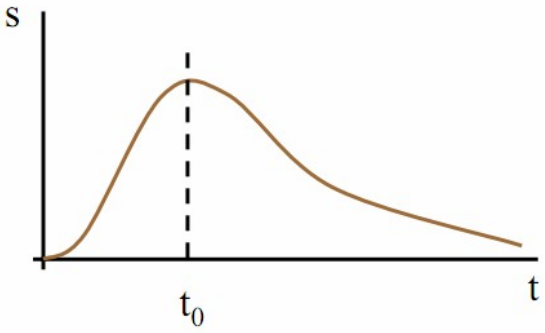


Rozložení řešitelské kapacity v čase

Model rozložení aproximován pomocí vlny s(t) (Rayleigh)

s(t) - počet řešitelů v čase *t*

$$s(t) = T\frac{t}{t_0}e^{\frac{-t^2}{2t_0}}$$



Celková spotřeba práce T:

$$T = \int_0^\infty s(t)dt$$

Atd… myslím, že další grafy zobrazovat nemá moc smysl… pro zvědavé nechť nahlédnout na přednášky.

Ukončení projektu

Význam

- určit, co se dělalo dobře, co bylo špatně
- co fungovalo a co ne
- jak to udělat příště lépe

Hlavním cílem není „pomoci“ pitvanému projektu, ale organizaci. Analýza je potřebná pro pochopení výkonnosti procesu při řešení daného projektu, ze které získáme poznatky o schopnostech samotného procesu.

Zpráva o závěrečné analýze

Obecné informace a informace vztažené k procesu

Souhrnná informace o projektu, dosažená produktivita a kvalita, použitý proces a jeho odchylky, odhady a aktuální časy, použité nástroje apod.

Rizikové řízení

Odhadnutá rizika včetně plánovaných opatření. Skutečná rizika a jejich vyřešení.

Velikost

Odhady pro jednoduché, středně složité a složité moduly. Sjednocení velikosti podle jedné metriky

(např. SLOC ⇒ FP).

Práce

Odhad práce a skutečná práce. Práce zhodnocená podle etap. Cena práce věnované kvalitě (procento práce věnované přezkoumání, testování, přepracování, projektově zaměřené školení).

Defekty

Souhrn nalezených defektů s analýzou významnosti, etap detekce, etap zavedení, efektivita odstranění defektů.

Aktiva procesu a dodatky

Jiné užitečné artefakty procesu, které mohou být potenciálně přínosné pro budoucí projekty.

Kauzální analýza

Po ukončení projektu známe výkonnost procesu. Pokud vybočuje výkon z běžných mezí, pak hledáme příčiny. Kauzální analýza zkoumá velké odchylky a identifikuje jejich příčiny, obvykle pomocí diskuse a brainstormingu.

Obecné informace

Kód projektu	...
Životní cyklus	Vývoj, úplný životní cyklus
Oblast	Finance. Webová aplikace pro správu účtů.
Vedoucí projektu/vedoucí modulu	...
Obchodní manažer	...
Konzultant kvality SW	...

Shrnutí výkonu

Parametr	**Aktuální**	**Odhad**	**Odchylka**	**Důvod velké odchylky**
Práce (pracovník/den)	597	501	19%	2 velké požadavky na změnu
Vrchol počtu pracovníků	9	9	0	N/A
Počátek	03/04/00	03/04/00	0	N/A
Konec	03/11/00	30/11/00	27 dní	změny 5% práce navíc
Kvalita (#defektů/FP)	0.02	0.0125		prevence a inkrem. proces
Produktivita	58	57	2%	N/A
Cena kvality	31.4%	33%	5%	N/A
Rychlost zanášení defektů	0.022	0.03	-26%	zlepšení díky prevenci chyb
Efektivita odstranění defektů	97.4	97	malá	N/A

Detaily procesu

Přizpůsobení procesu	použit Rational Unified Process
	analýza a vývoj prováděny iterativně - 2 iterace na analýzu a návrh, 3 iterace na vývoj
	sledování požadavků bylo řešeno pomocí nástroje Requsite Pro

Použité nástroje

Poznámky o použitých nástrojích	Externí nástroje: VSS, VJA, Requsite Pro, MSP
	Interní nástroje: BugsBunny, WAR

Řízení rizik

Rizika identifikovaná na začátku projektu	
Riziko 1	nedostatek podpory DB architekta a DB administrátora - zaměstnanců zákazníka
Riziko 2	nesprávné použití RUP, protože se jedná o první nasazení
Riziko 3	odchod zaměstnanců
Riziko 4	problémy s přístupem k zákazníkově DB pomocí sítě

Rizika identifikovaná během projektu	
Riziko 1	vliv konverze na VAJ 3.0
Riziko 2	nedostatek podpory DB architekta a DB administrátora - zaměstnanců zákazníka
Riziko 3	nesprávné použití RUP, protože se jedná o první nasazení
Riziko 4	odchod zaměstnanců

Řízení rizik - poznámky ke snížení vlivu

1. **Riziko 1:** Jasná formulace rizika přispěla k získání souhlasu zákazníka s odložením konverze a upřesněním nákladů souvisejících s konverzí.
2. **Riziko 2:** Uspěla strategie pečlivého a včasného plánování a nasazení koordinátora v místě u zákazníka.
3. **Riziko 3:** Školení řešitelského týmu v RUP bylo úspěšné.
4. **Riziko 4:** Riziko zůstalo, ale neuskutečnilo se. Vliv by byl minimální, neboť více řešitelů bylo informováno o každé kritické činnosti

Velikost

	Odhad	Skutečnost
Počet jednoduchých PU	5	5
Počet středně složitých PU	9	9
Počet složitých PU	12	12

Poznámky k odhadům

Klasifikační kritéria: Standardní definice složitosti PU vyhověla.

Velikost systému v FP: Velikost změřená v LOC byla normalizována na FP s použitím konverze. Java: 21 LOC = 1 FP, COBOL: 107 LOC = 1 FP

	Velikost v LOC	Velikost v FP
Java	33865	1612
COBOL	1241	12

Plán

Etapa	skutečná doba (dny)	odhadnutá doba (dny)	% skluz	důvody skluzu
Požadavky	28,67	31	-6,5	
Celkový návrh	0	0	0,0	
Podrobný návrh	38,8	42	-6,7	
Kódování	132	135	-1,6	
Testování jednotek	9	10	-9,3	
Vývoj celkem	141	144	-2,1	
Integrační testování	40	40	0,0	
Testování systému	15	0	0,0	
Předávací testy	30	10	200,0	prodlouženo na žádost zákazníka

Práce

Etapa	úkol	přezkoumání	přepracování	celkem
Požadavky	210.0	10.0	60.0	280.0
Celkový návrh	0.0	0.0	0.0	0.0
Podrobný návrh	652.0	14.0	29.5	695.5
Kódování	1188.0	39.5	76.5	1304.0
Testování jednotek	129.5	0.0	17.0	146.5
Integrační testování	567.5	6.0	160.5	734.0
Testování systému	90.0	0.0	0.0	90.0
Předávací testy	336.5	0.0	0.0	336.5
Životní cyklus celkem	3173.5	69.5	343.5	3586.5

	úkol	přezkoumání	přepracování	celkem
Řízení projektu	733.1	0.0	0.0	733.1
Školení	104.5	0.0	0.0	104.5
Řízení konfigurací	317.0	0.0	0.0	317.0
Různé	488.5	0.0	0.0	488.5
Řízení celkem	1643.0	0.0	0.0	1643.0

Celková práce (osoby-hodiny)	4816.50	69.50	343.50	5229.50
Celková práce (osoby-měsíce)	25.76	0.37	1.84	27.97

Cena kvality

COQ

= (přezkoumání + přepracování + testování + školení) x 100 / práce celkem

= (69.5 + 343.5 + 129.5 + 567.5 + 90 + 336.5 + 104.5) x 100 / 5229.5

= 31.4 %

Defekty

Etapa	skutečný #defektů	% z celk. #defektů	odhad #defektů	% z celk. odhadu #defektů	% odchylky
Přezk.požadavků a návrhu	11	10	29	20	-62
Přezkoumání kódu	58	50	29	20	100
Testování jednotek	15	13	57	40	-73
Integrační a systémové testování	29	25	25	17	16
Předávací testy	3	2	5	3	-40
Celkem	116	100	145	100	-20

Zdůvodnění odchylek ...

Zdůvodnění odchylek:

1. Prevence defektů snížila rychlost vzniku defektů v pozdějších etapách, došlo k celkovému snížení rychlosti zanášení defektů.

2. U předchozího projektu, který byl základem pro odhad, se dělalo méně přezkoumání. Důraz byl kladen na testování jednotek. U současného projektu bylo nalezeno více defektů při inspekcích a počet defektů propuštěných do testování jednotek výrazně poklesl.

Efektivita odstraňování defektů

Etapa detekce defektu	Etapa zanesení defektu			Efektivita odstranění defektu
	Požadavky	Výroba	Návrh	
Přezkoumání požadavků	5			100%
Přezkoumání návrhu	0	6		100%
Přezkoumání kódu	0	0	58	55% 58/(58+15+29+3)
Testování jednotek	0	0	15	32% 15/(15+29+3)
Integrační a systémové testování	0	0	29	91% 29/(29+3)
Předávací testy	0	0	3	100%

Celková efektivita odstraňování defektů: = 113/116 = 97,4%

Rozložení defektů podle závažnosti

	závažnost	#defektů	% z celk. #defektů
1.	kosmetická	26	22.4
2.	malá	51	44.0
3.	velká	36	31.0
4.	kritická	3	2.6
5.	ostatní	---	---
	celkem	116	

Rozložení defektu podle typu

	Typ defektu	#defektů	% z celk. #defektů
1.	logika	33	28.4
2.	standardy	29	25.0
3.	výkonnost	24	20.7
4.	nadbytečný kód	14	12.0
5.	uživatelské rozhraní	9	7.7
6.	architektura	4	3.5
7.	konzistence	2	1.7
8.	znovupoužitelnost	1	0.9

Kauzální analýza a získané poznatky

- Zhodnocení projektu ve vztahu k plánování:
- Bylo jen velmi málo výrazných odchylek od očekávaných parametrů procesu.
- Zdůvodnění odchylek je zapsáno u všech velkých odchylek.

Nejdůležitější získané zkušenosti jsou:

1. Inkrementální vývoj je velmi užitečný pro dosažení vyšší kvality a produktivity, protože ....
2. Prevence defektů výrazně redukuje rychlost vnášení defektů. Z pohledu spotřebované práce přináší prevence 5 až 10 násobnou úsporu při redukci práce vynaložené na předělání produktu.
3. Diskuse se zákazníkem o vlivu velkých změn ....
4. Efektivita přezkoumání kódu a testování jednotek je zatím nízká. Je nutné prověřit procesy ....

Aktiva procesu

- plán řízení projektu
- harmonogram projektu
- plán řízení konfigurací
- kódovací standardy Java
- kontrolní seznam pro přezkoumání kódu

- kontrolní seznam pro přezkoumání integračního plánu
- zprávy z kauzální analýzy pro prevenci chyb

Artefakty, které vznikly při procesu a mohou být užitečné i pro jiné projekty.