Řízení kvality SW produktů

Klasický pohled na kvalitu SW

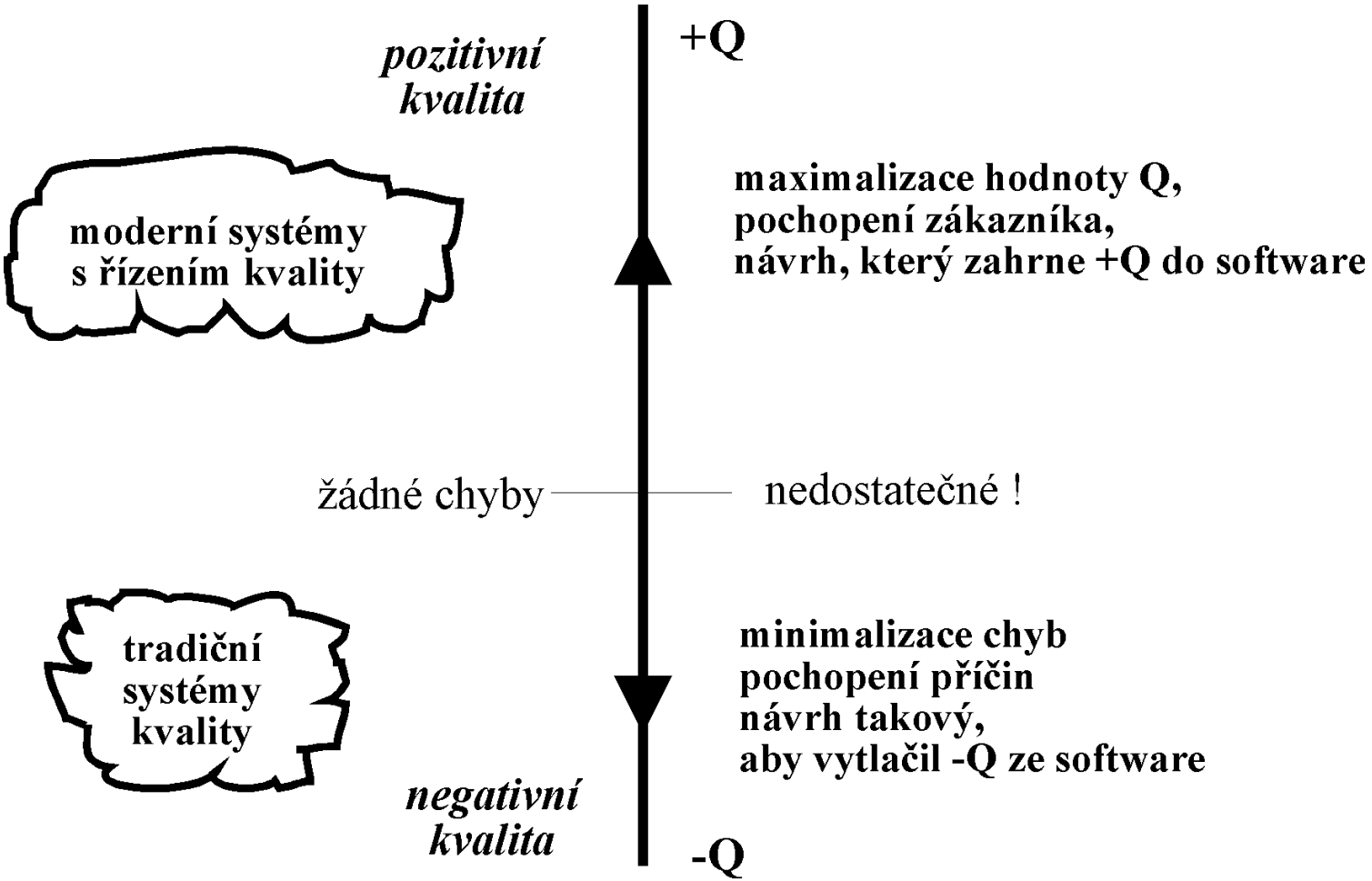
Každý program dělá něco správně; nemusí však dělat to, co chceme, aby dělal.

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.*

Přímo měřitelné faktory

- #chyb/KLOC/čas

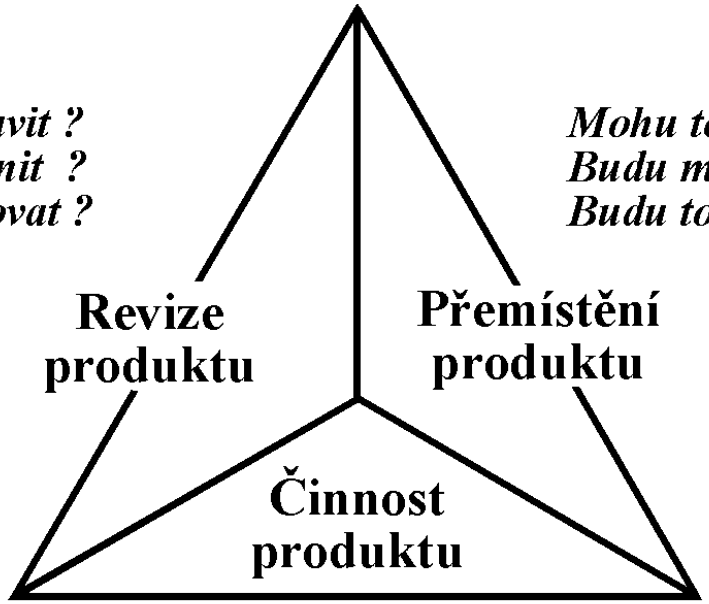
Pouze nepřímo měřitelné faktory

- použitelnost, udržovatelnost

Kategorie faktorů kvality:

- operační charakteristiky
- schopnost akceptovat změny
- adaptibilita na nové prostředí

Faktory kvality - McCall

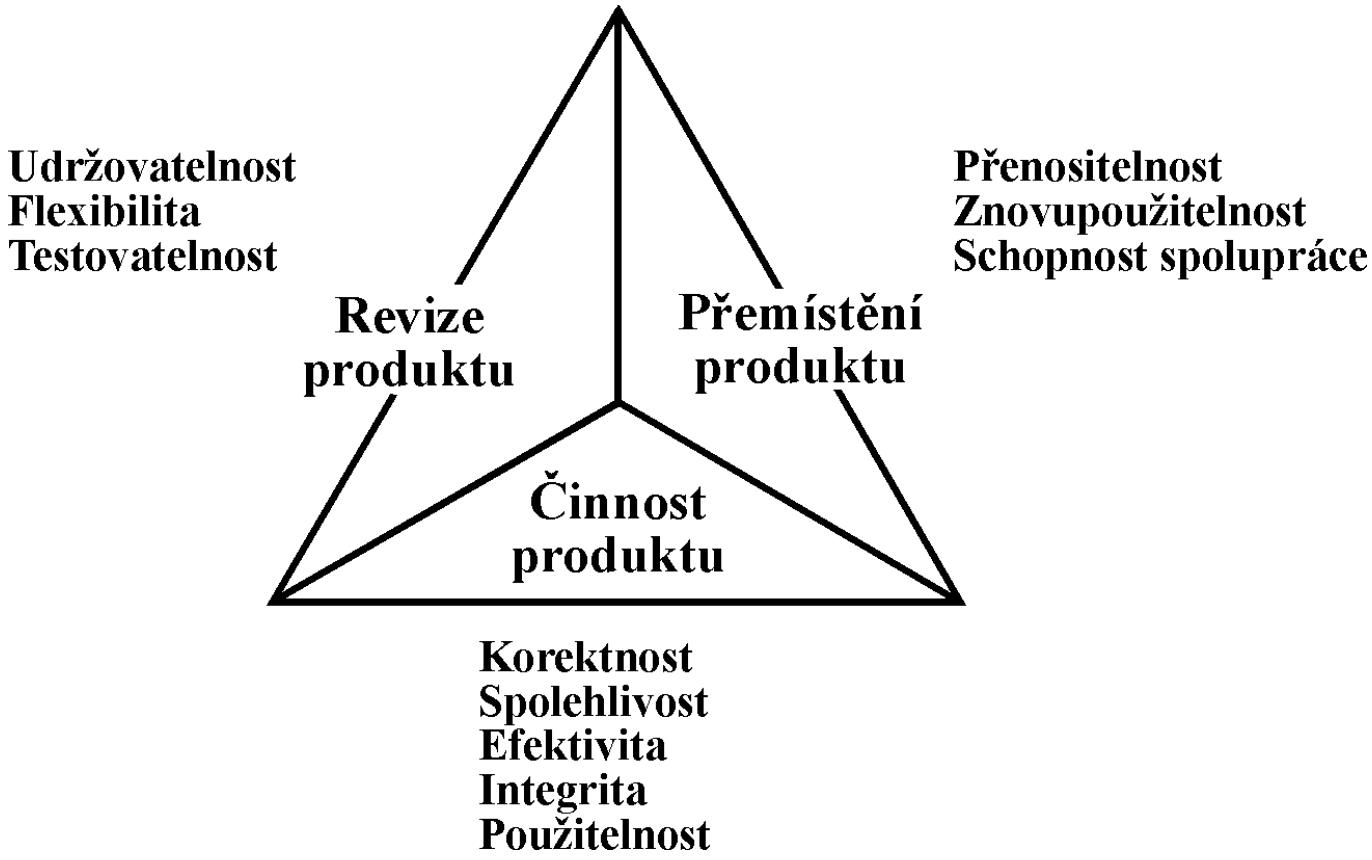


Mohu to opravit ?
Mohu to změnit ?
Mohu to testovat ?

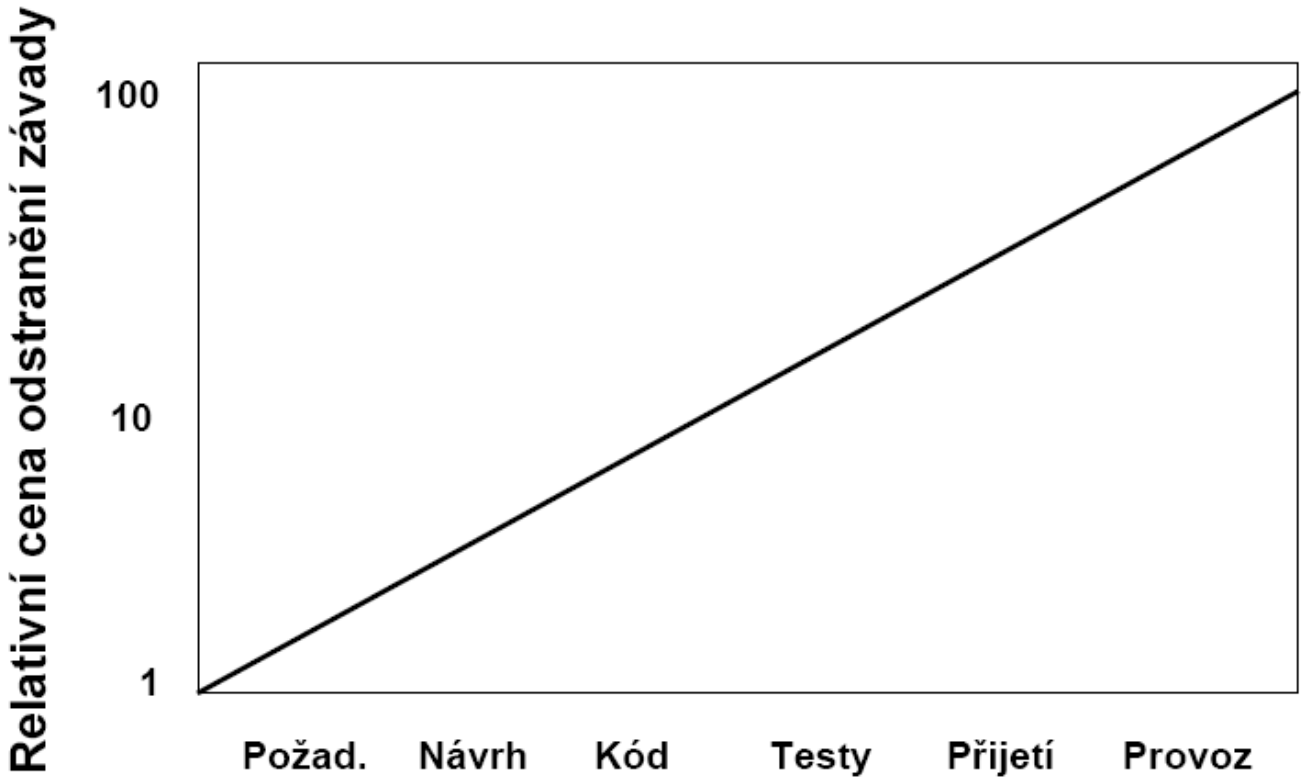
Mohu to použít na jiném počítači ?
Budu moci znovu použít část software ?
Budu to moci připojit k jinému systému ?

Dělá to, co chci ?
Dělá to, co má, přesně celou dobu ?
Poběží to na počítači tak dobře, jak jen lze ?
Je to bezpečné ?
Je to navrženo pro uživatele ?

Faktory kvality - McCall



Relativní cena odstranění závady



Zdroj: Barry W. Boehm, 1981, COCOMO

IBM ortogonální klasifikace defektů (ODC)

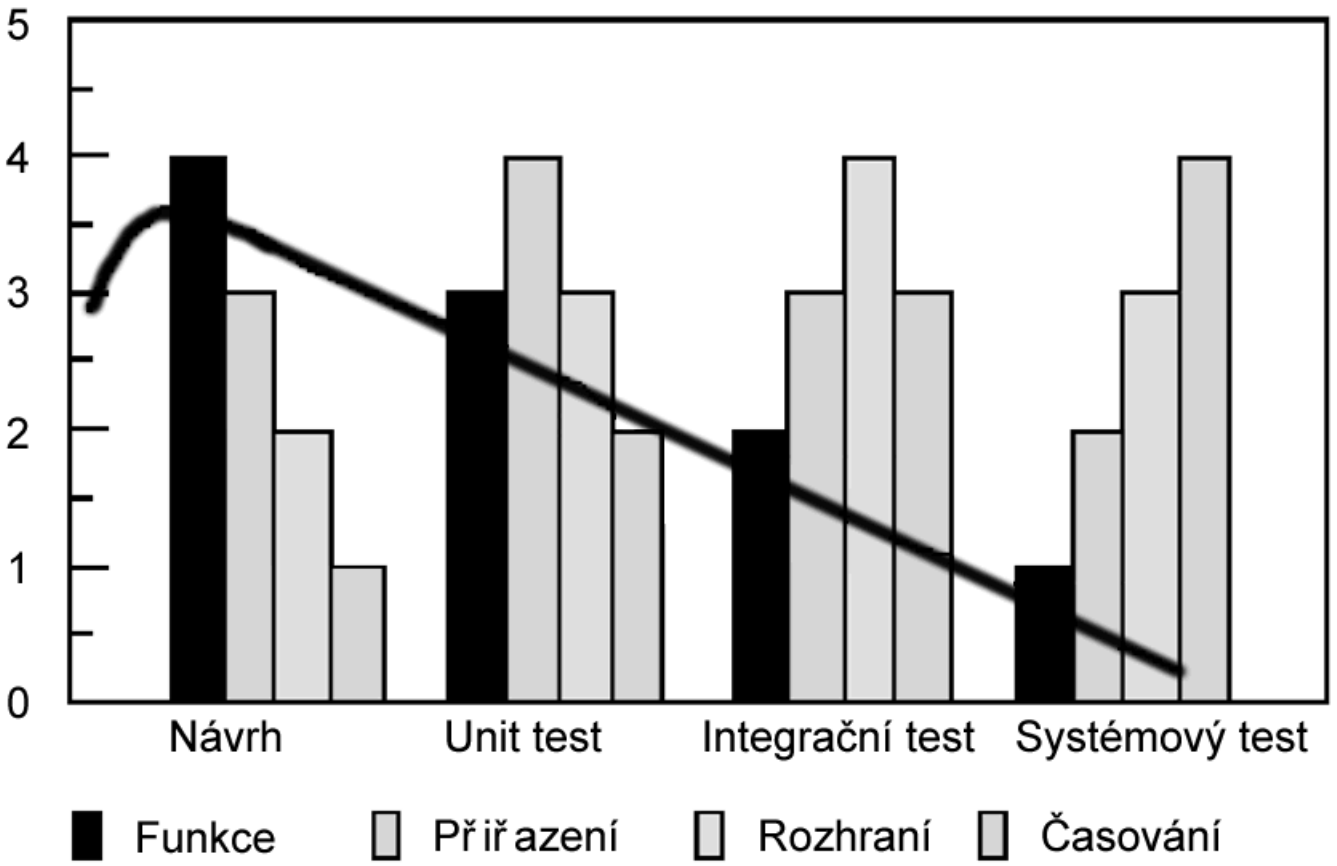
- Funkce - chyba ovlivňující schopnosti, rozhraní uživatelů, rozhraní výrobku, rozhraní s HW architekturou nebo globální datovou strukturou.
- Rozhraní - chyba při interakci s ostatními komponentami nebo ovladači přes volání, makra, řídící bloky nebo seznamy parametrů.
- Ověřování - chyba v logice programu, která selže při validaci dat a hodnot před tím, než jsou použity.
- Přiřazení - chyba při inicializaci datové struktury nebo bloku kódu.
- Časování/serializace - chyba, která zahrnuje časování sdílených a RT prostředků.
- Sestavení/balení/spojování - chyba související s problémy s repozitory projektu, změnami vedení, nebo správou verzí.
- Dokumentace - chyba, která ovlivňuje publikace a návody pro údržbu.
- Algoritmus - chyba, která se týká efektivity nebo správnosti algoritmu nebo datové struktury, ne však jejich návrhu.

Typ defektu a asociace s etapou

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

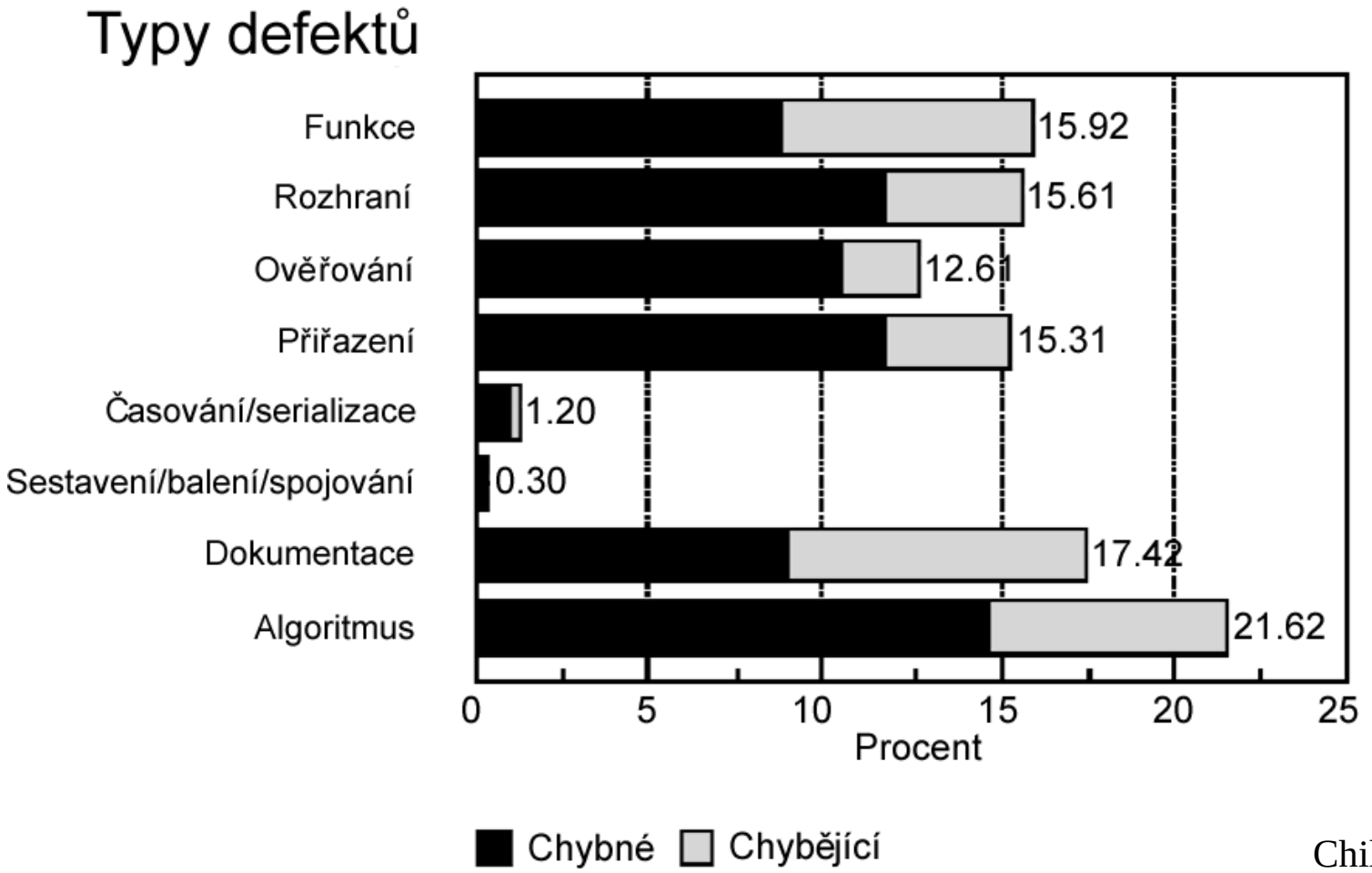
Chillarege et al

Funkční selhání podle etapy



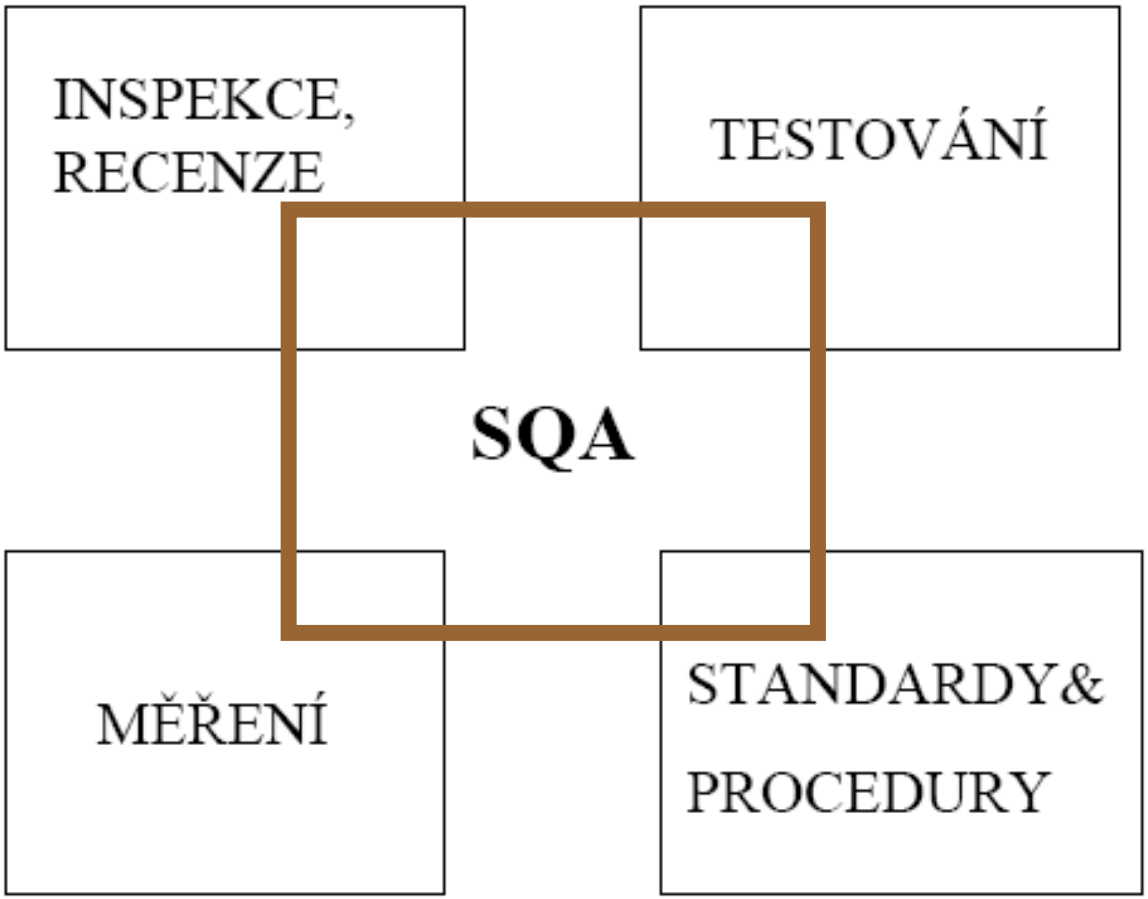
Chillarege et al

Rozložení druhů chyb při testování

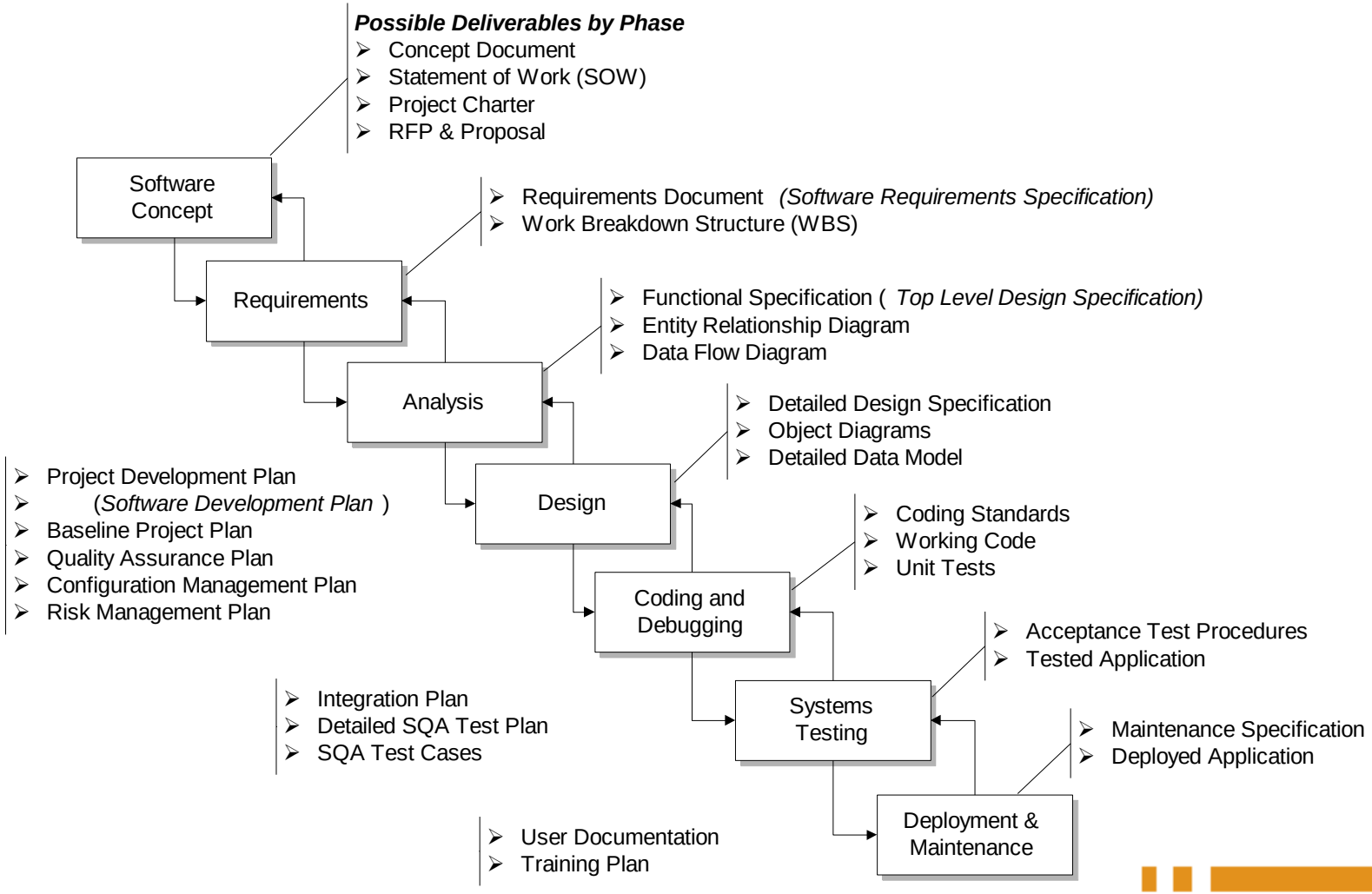


Chillarege et al

SQA - Software Quality Assurance



Produkty podle etap



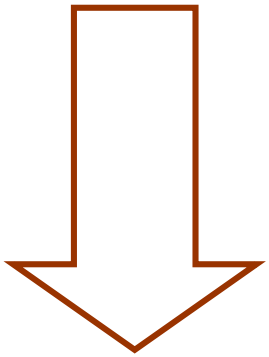
Přínos SQA

Etapa	Cena nalezení a opravy
Požadavky	0.75
Návrh	1.0
Kódování	1.5
Testování	3.0
Systémové testy	10.0
Provoz	60-100.0

Pozn.: Cena normalizovaná vzhledem k ceně v etapě návrhu

Přezkoušení SW

Méně formální



Více formální

- Konverzace
- Přezkoušení mezi spolupracovníky
- Neformální prezentace
- Formální prezentace
- Prohlídka
- Recenze, inspekce

- V literatuře je argumentováno (např. Pressman), že efektivita roste se zvyšující se formálností.
- Probíhá v období tvorby kódu, odstranění nalezených nedostatků je relativně levné.

Typy formálního přezkoušení

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

Cíle formálního přezkoušení

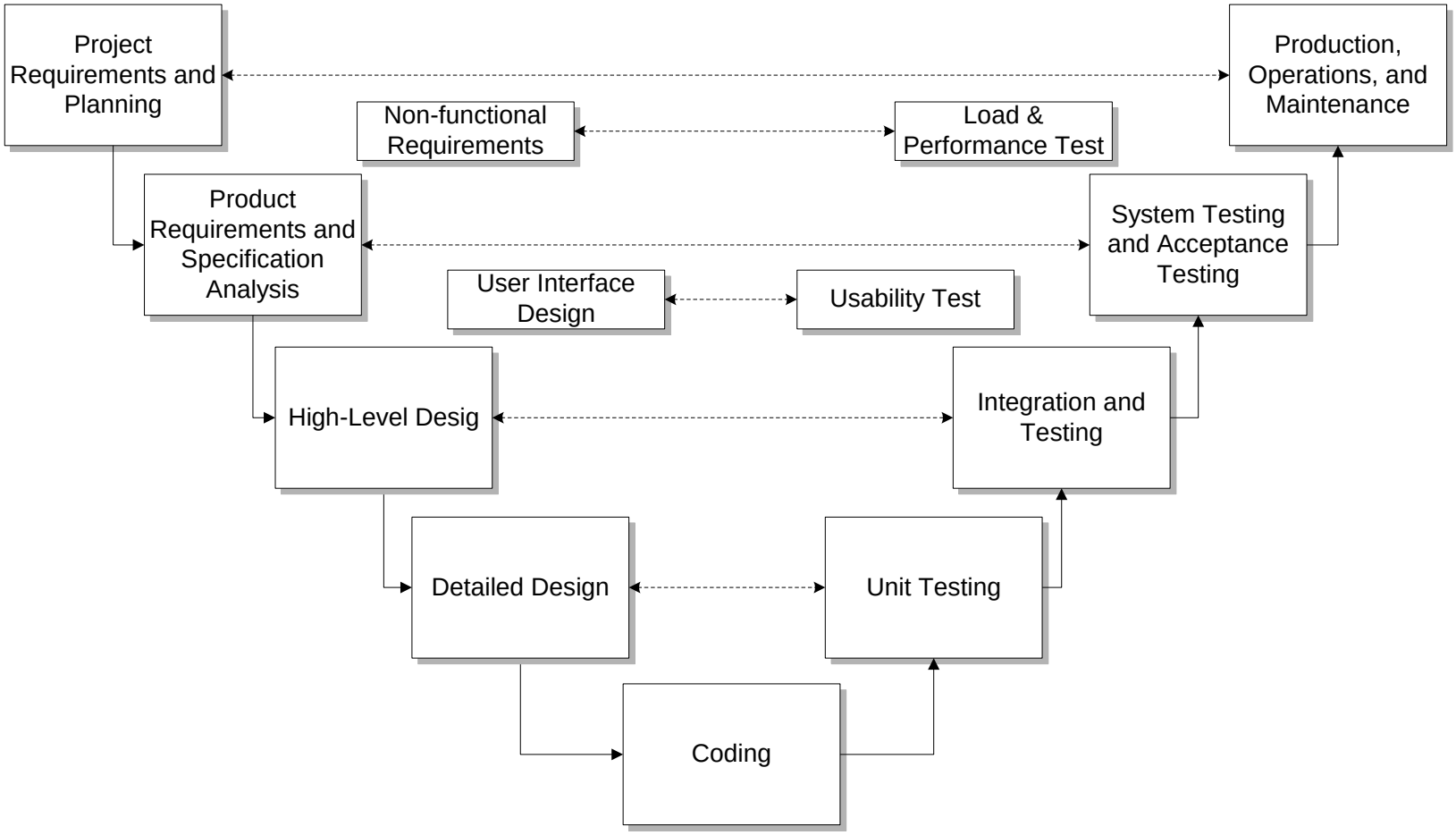
- Odhalit chyby ve funkci, logice a implementaci software.
- Ověřit, že zkoumaná položka splňuje požadavky (odpovídá požadavkům).
- Zajistit, že položka byla prezentována s použitím předdefinovaných standardů.
- Zajistit jednotný vývoj.
- Zvýšit řiditelnost projektů.

Co je testování?

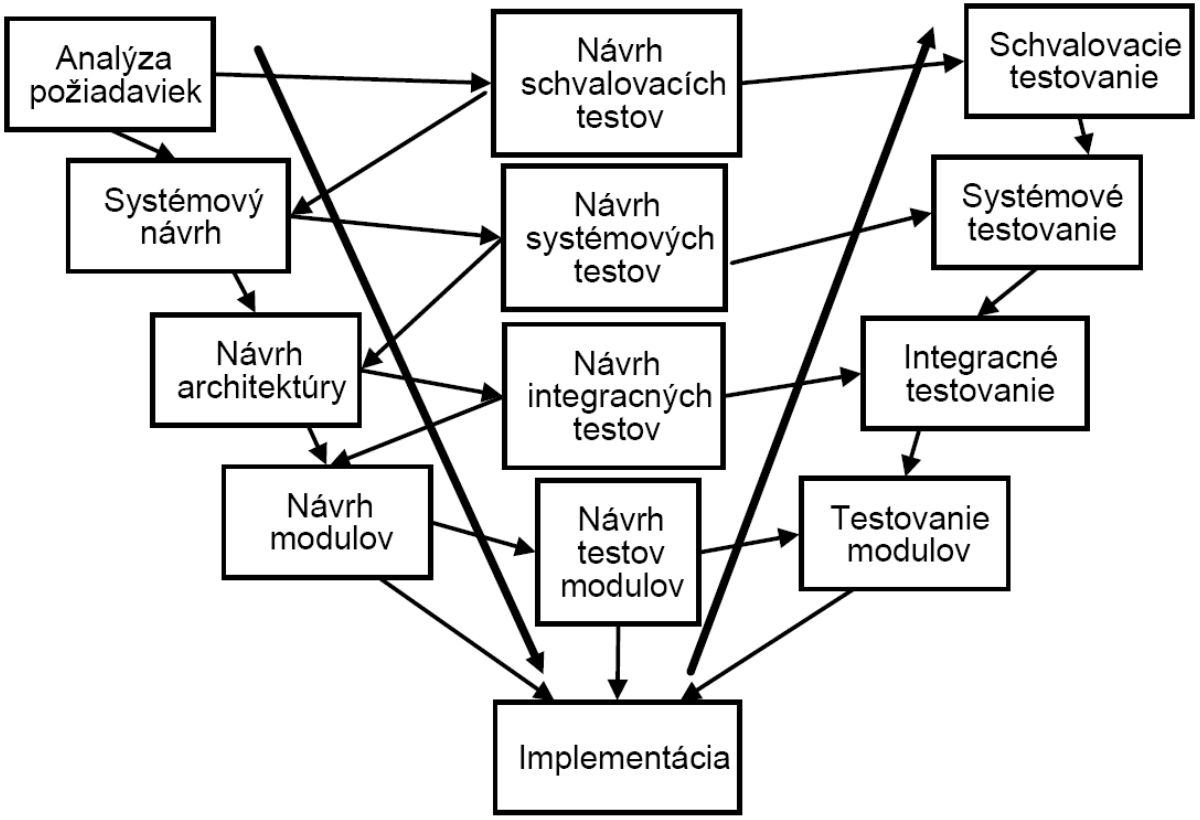
- Testování je proces spuštění programu s cílem nalézt chyby.
- Dobrý testovací případ má vysokou pravděpodobnost nalezení dosud nenalezené chyby.
- Úspěšný test je takový, který odhalí dosud neodhalenou chybu.

- Myers

V - procesní model



V - procesní model



Co testování ukazuje?

- Testování nemůže ukázat nepřítomnost defektů, může pouze ukázat, že v softwaru jsou chyby.
- Testování také ukazuje funkce a výkon.
- A je také ukazatelem kvality software.

Verifikace &Validace

Každý inženýrský výrobek může být testován dvěma způsoby:

- test proti specifikovaným funkcím = Validace

  “Dělat správné věci”

- test proti vnitřní činnosti = Verifikace

  “Dělat věci správně”

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.

Tým kvality ⟷ Realizační tým

„Černé a bílé skříňky“

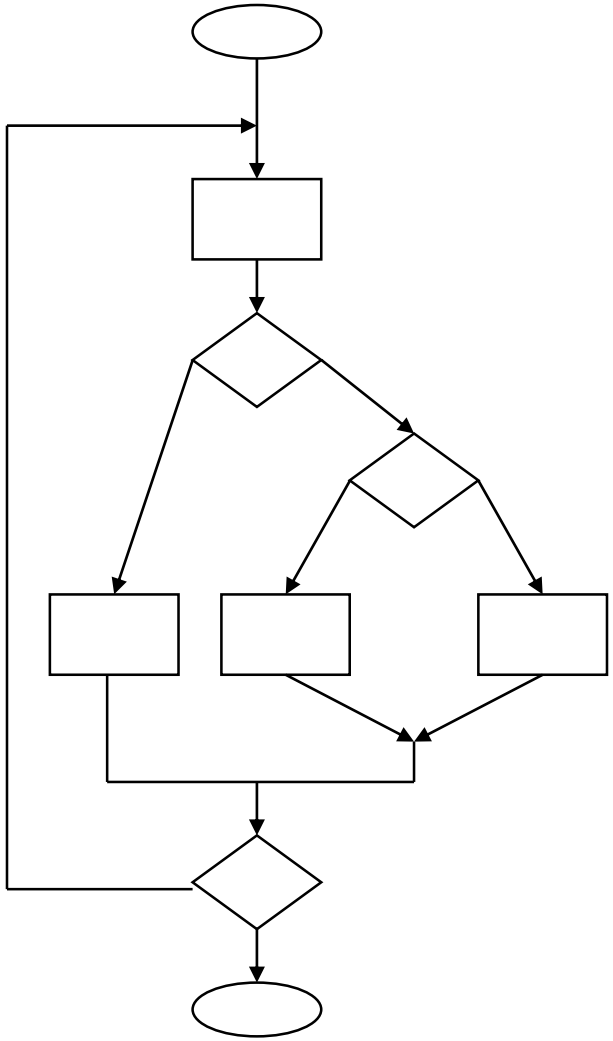
FUNKCE
- test činnosti každé funkce
- test ČERNÉ SKŘÍŇKY

VNITŘNÍ PRÁCE
- test, zda ‘všechny motory pracují’
- test BÍLÉ SKŘÍŇKY

Testování „bílá skříňka“

- Zohledňuje strukturu programu
- Pokrývá
  - provedené příkazy
  - cesty průchodu kódem

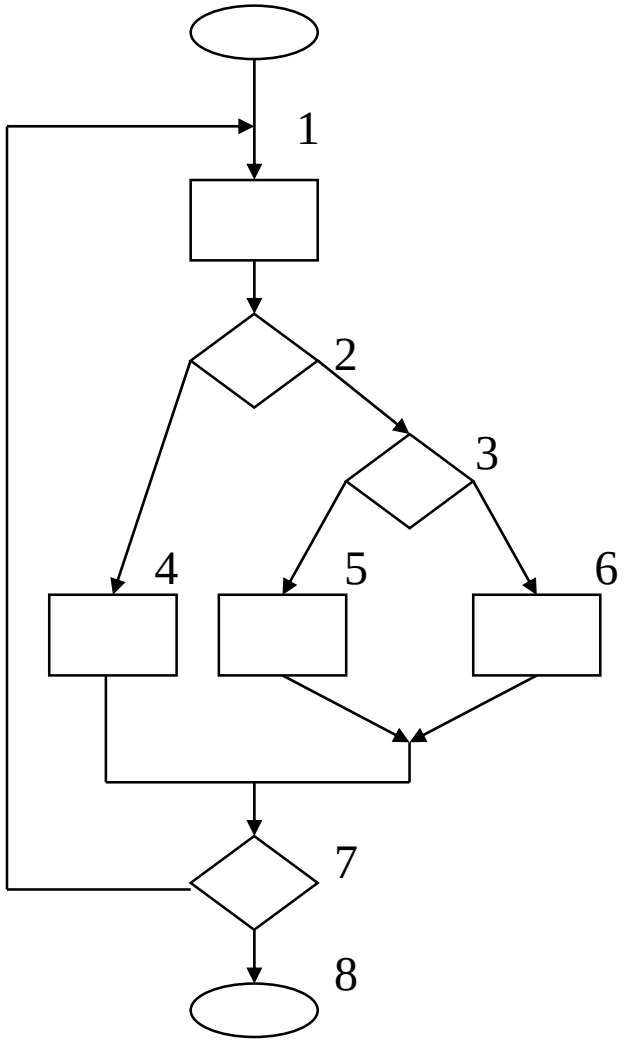
Testování „bílá skříňka“



1. výpočet cyklomatické složitosti:

   počet rozhodnutí + 1
   (predikátové uzly)
   nebo
   počet ploch (oblastí)
   nebo
   hrany – uzly + 2

Testování „bílá skříňka“



2. Nalezneme nezávislé cesty.

   Protože cyklomat. složitost = 4
   existují 4 nezávislé cesty:

   cesta 1: 1,2,3,6,7,8
   cesta 2: 1,2,3,5,7,8
   cesta 3: 1,2,4,7,8
   cesta 4: 1,2,4,7,1,2,4,...7,8

Testování jednotek, modulů

- Typ testování „bílá skříňka“
  - někdy ale jako „černá skříňka“
- Kdo testuje jednotky?
  - vývojáři
  - testy jednotek jsou programovány
    - stejný jazyk jako moduly
    - alt.název “Testovací drivery”
- Individuální testy mohou být seskupeny
  - „Kolekce testů“ (Test suites)
- Kdy se testují jednotky?
  - postupně během vývoje
  - po dokončení individuálních modulů

Integrace & Testování

- Vývoj/integrace/testování
  - nejčastější místo, kde dochází k překrývání aktivit
- Někdy je integrace/testování považováno za jednu etapu
- Postupně propojuje funkcionalitu
- QA tým pracuje souběžně s vývojovým týmem

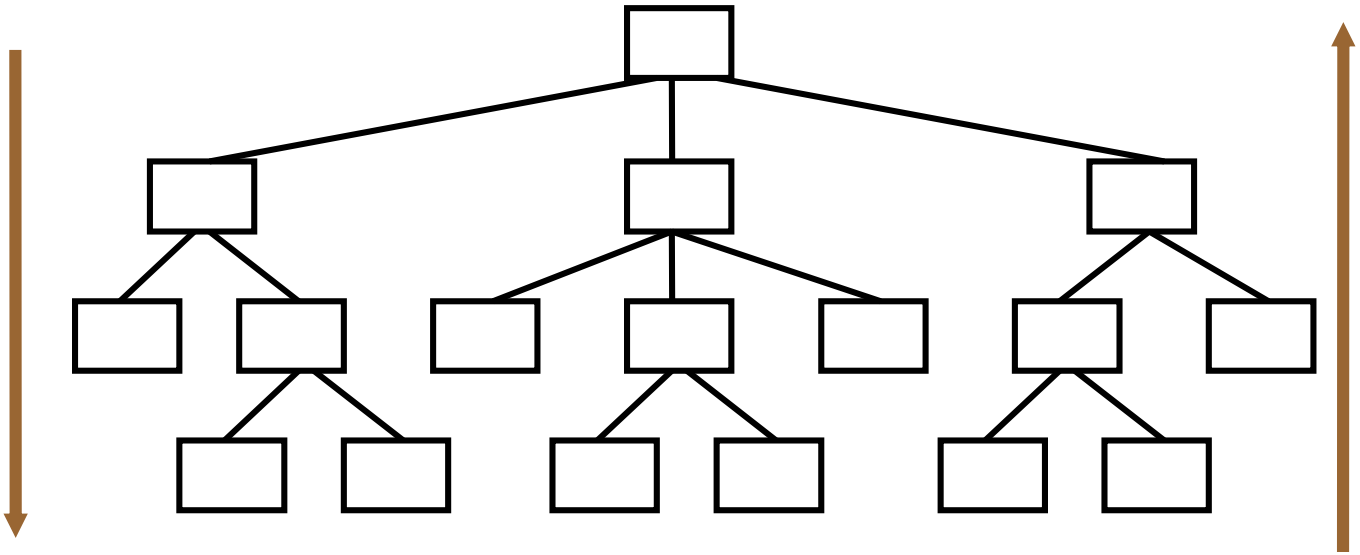
Shora dolů

- Nejprve je implementováno jádro (kostra) systému.
- Zkombinováno do minimální „skořápky“ systému.
- Pro doplnění neúplných částí se použijí „protézy“ nahrazované postupně aktuálními moduly.

Zdola nahoru

- Začne s individuálními moduly a sestavuje zdola.
- Individuální jednotky (po testování jednotek) jsou kombinovány do subsystémů.
- Subsystémy jsou kombinovány do celku.

Testování shora-dolů, zdola-nahoru

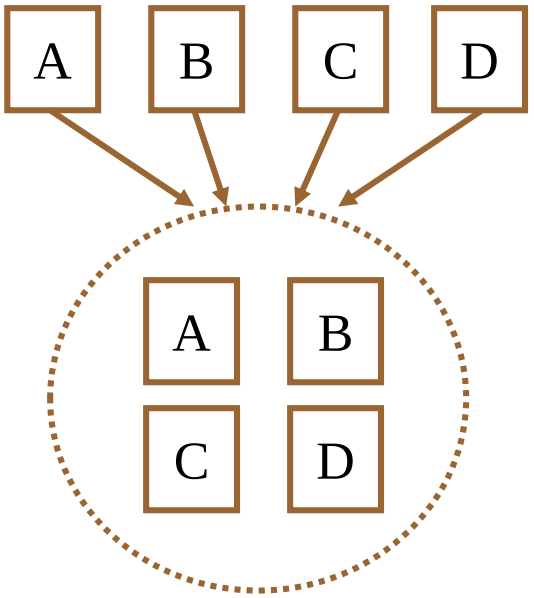


Shora-dolů (TDT): použití „stubs“ (pahýly, protézy) - jednoduché náhražkové objekty se shodným rozhraním.

Zdola-nahoru(BUT): klasický testovací proces s nadřazenými testovacími objekty - „drivers“.

Testování shora-dolů odhaluje chyby analýzy a návrhu, je v souladu s prototypováním.

Integrační testování



Testování modulů

Integrační testování

Kde je chyba ?

⇩

Inkrementální integrace a testování

Klasifikace metrik

- Metriky produktu
  - Explicitní výsledky vývoje SW
  - Kód, výstupy, moduly, dokumentace …

- Metriky procesu
  - Činnosti spojený s vývojem SW

- Metriky zdrojů
  - Zdroje (vstupy) procesu vývoje SW
  - Hardware, znalosti, lidé

Měření složitosti

- LOC – velikost kódu je funkcí složitosti SW
- Závisí na programovacím jazyce a schopnostech programátorů

- Halstead's Software Science
  - n_1: počet odlišných operátorů
  - n_2 : počet odlišných operandů
  - N_1: počet všech operátorů
  - N_2: počet všech operandů

Příklad

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

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

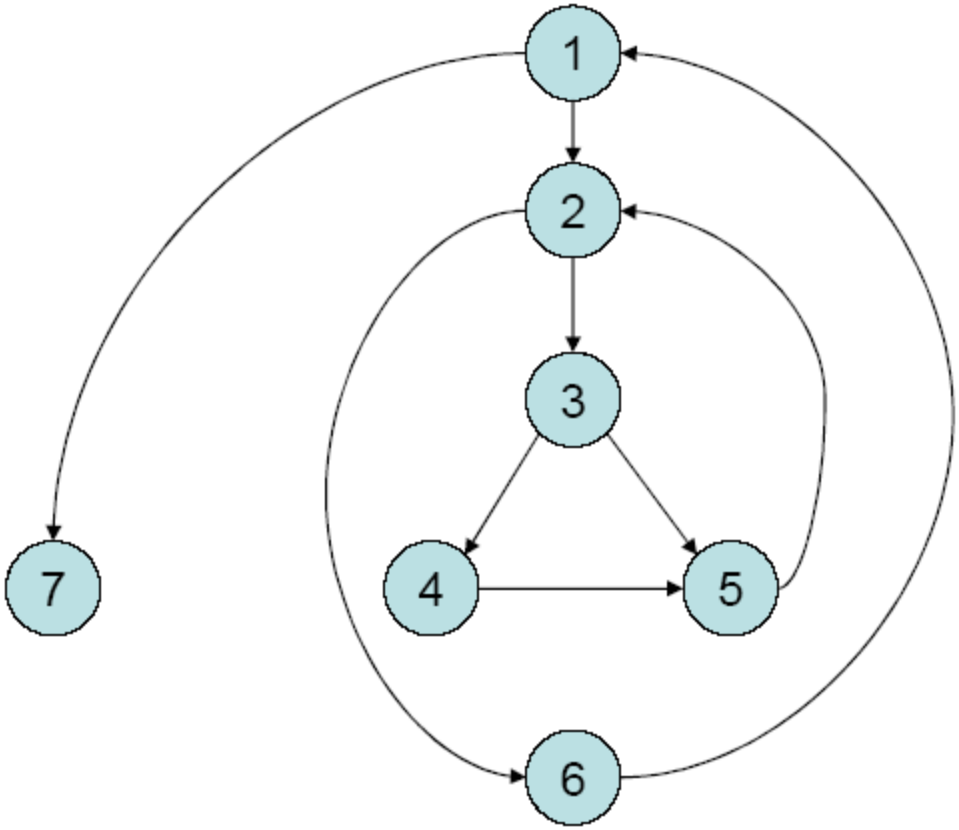
Halsteadovy metriky

- Délka: $N = N_1 + N_2$
- Slovník: $n = n_1 + n_2$
- Odhadnutá délka: $\tilde{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2$
  - Odhad pro dobře strukturované programy
- Poměr čistoty: $PR = \tilde{N} / N$

Cyklomatická složitost

- Množina nezávislých cest grafem
- $V(G) = E - N + 2$
  - E je počet hran
  - N je počet uzlů
- $V(G) = P + 1$
  - P je počet větvení (na 2 cesty)

Cyklomatická složitost - Flow Graph



Cyklomatická složitost

- $V(G)$ je počet oblastí grafu.
- Počet oblastí roste s počtem větvení a cyklů.
- Kvantitativní metrika obtížnosti testování.
- Indikátor spolehlivosti.
- Praxe ukazuje, že hodnota $V(G)$ by neměla překročit 10, jinak bude testování velmi složité.

Metriky návrhu

- Složitost struktur
- Složitost dat
- Složitost systému

- Složitost struktur $S(i)$ modulu i
  - $S(i) = f_{out}^2(i)$
  - $f_{out}(i)$ počet přímo podřízených (volaných) modulů

Metriky návrhu

- Složitost dat $D(i)$
  - $D(i) = v(i) / [f_{out}(i) +1]$
  - $v(i)$ množství vstupů a výstupů modulu i

- Složitost systému $C(i)$
  - $C(i) = S(i) + D(i)$

Připojení modulů

- Datové a řídící toky
  - d_i - vstupní datové toky do modulu
  - c_i - vstupní řídící toky do modulu
  - d_o - výstupní datové toky z modulu
  - c_o - výstupní řídící toky z modulu
- Globální struktury
  - g_d - globální datové proměnné
  - g_c - globální řídící proměnné
- Okolí
  - w - počet volaných modulů
  - r - počet modulů, které volají modul

Metriky propojení

$M_c = k/m \qquad (k = 1)$

$m = d_i + ac_i + d_o + bc_o + g_d + cg_c + w + r$

a, b, c, k jsou nastaveny na základě aktuálních dat (zkušeností)

Globální hodnocení kvality výroby SW

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

CMM - Capability Maturity Model

také SEI model (Software Engineering Institute, Carnegie-Mellon Univ.), revize 1993

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

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

Vztah mezi MBNQA a ISO 9001