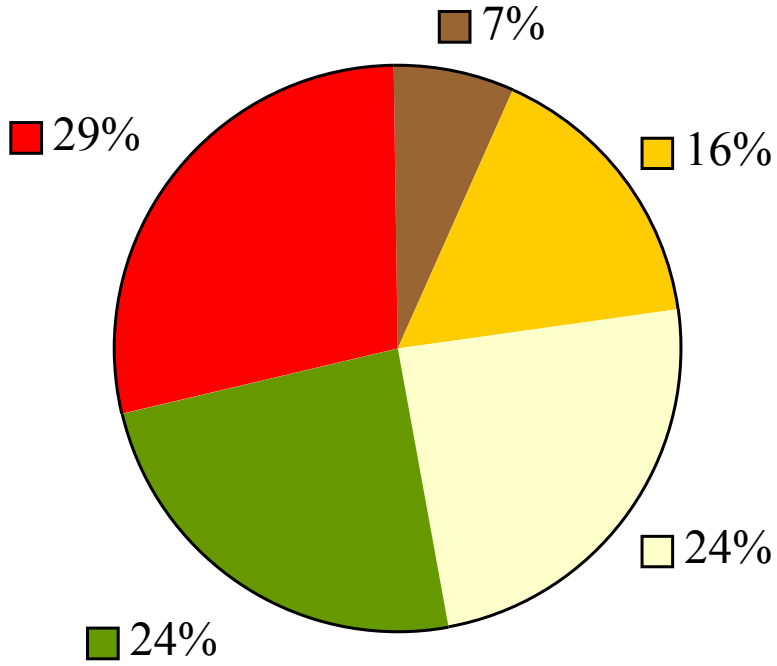
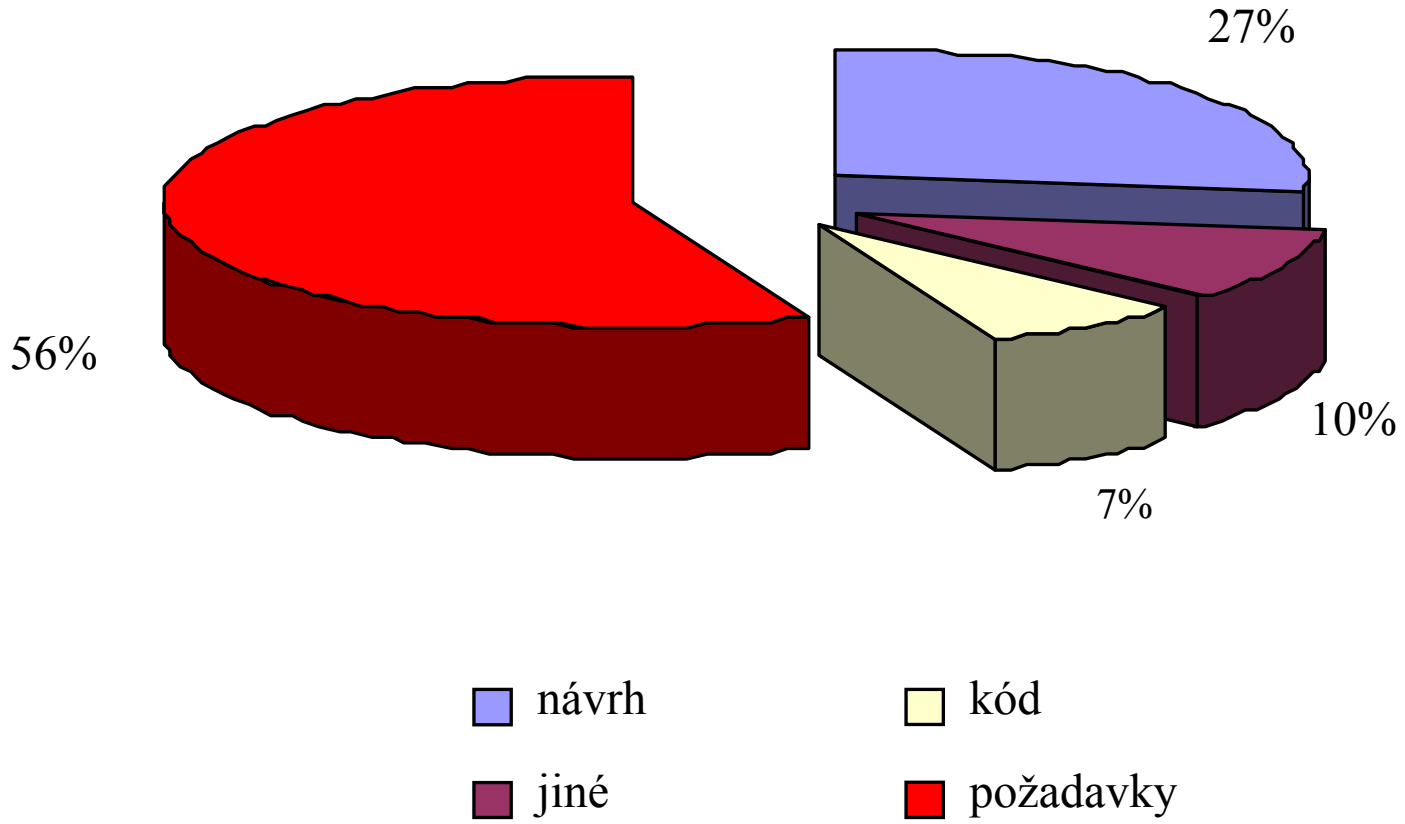
Testování SW produktů

Cena testování během vývoje



- požadavky
- předběžný návrh
- podrobný návrh
- testování kódu a jednotek
- integrační a systémové testy

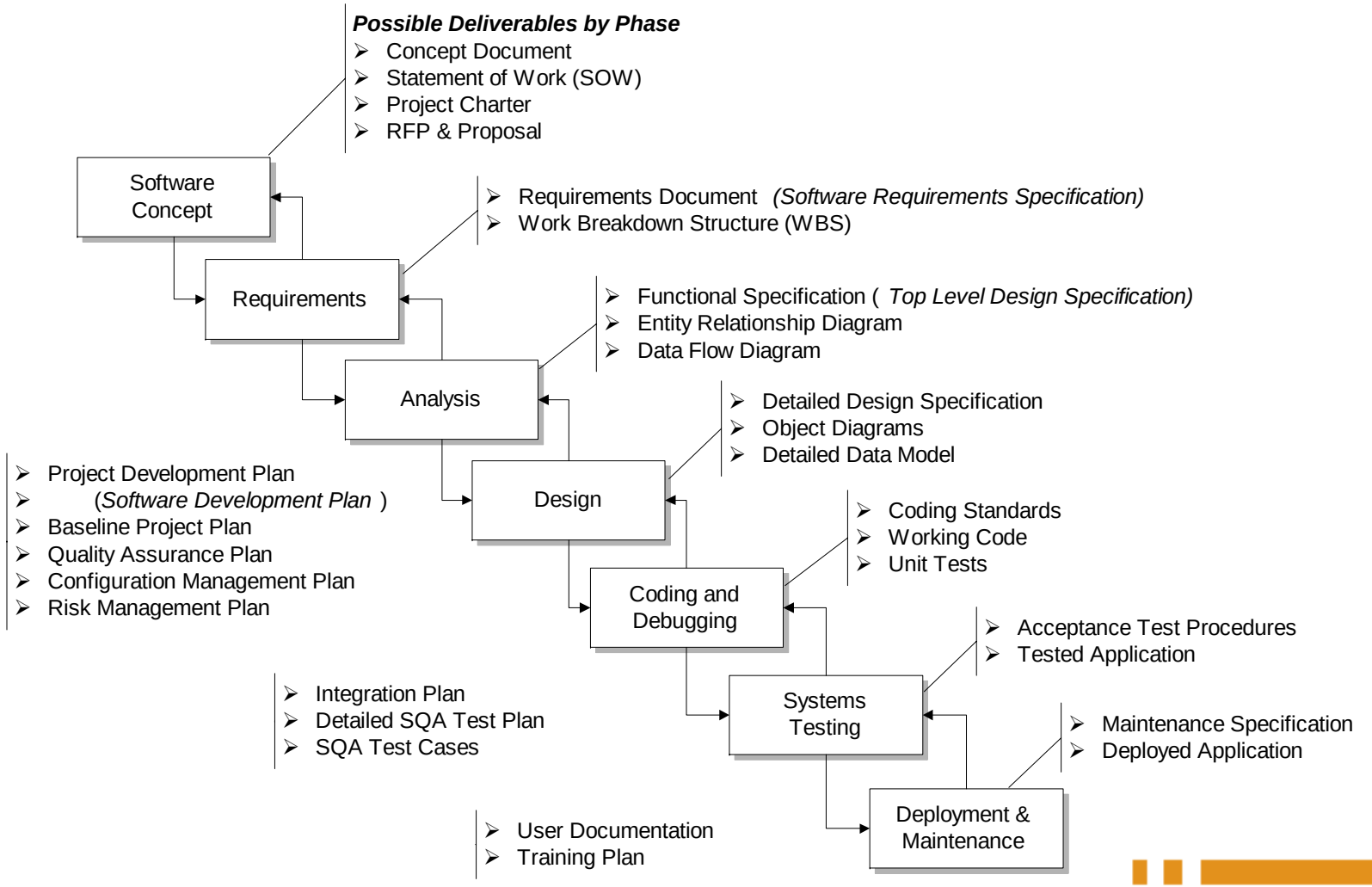
Zdroje defektů



Pokud by 99.9% funkcí bylo správně ...

- 9,703 šeků by bylo každou hodinu proplaceno z jiných bankovních účtů.
- 27,800 dopisů by se každou hodinu ztratilo.
- 3,000,000 nesprávných předpisů na léky by se ročně vydalo.
- 8,605 komerčních letů by každoročně během startu havarovalo.

Produkty podle etap



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

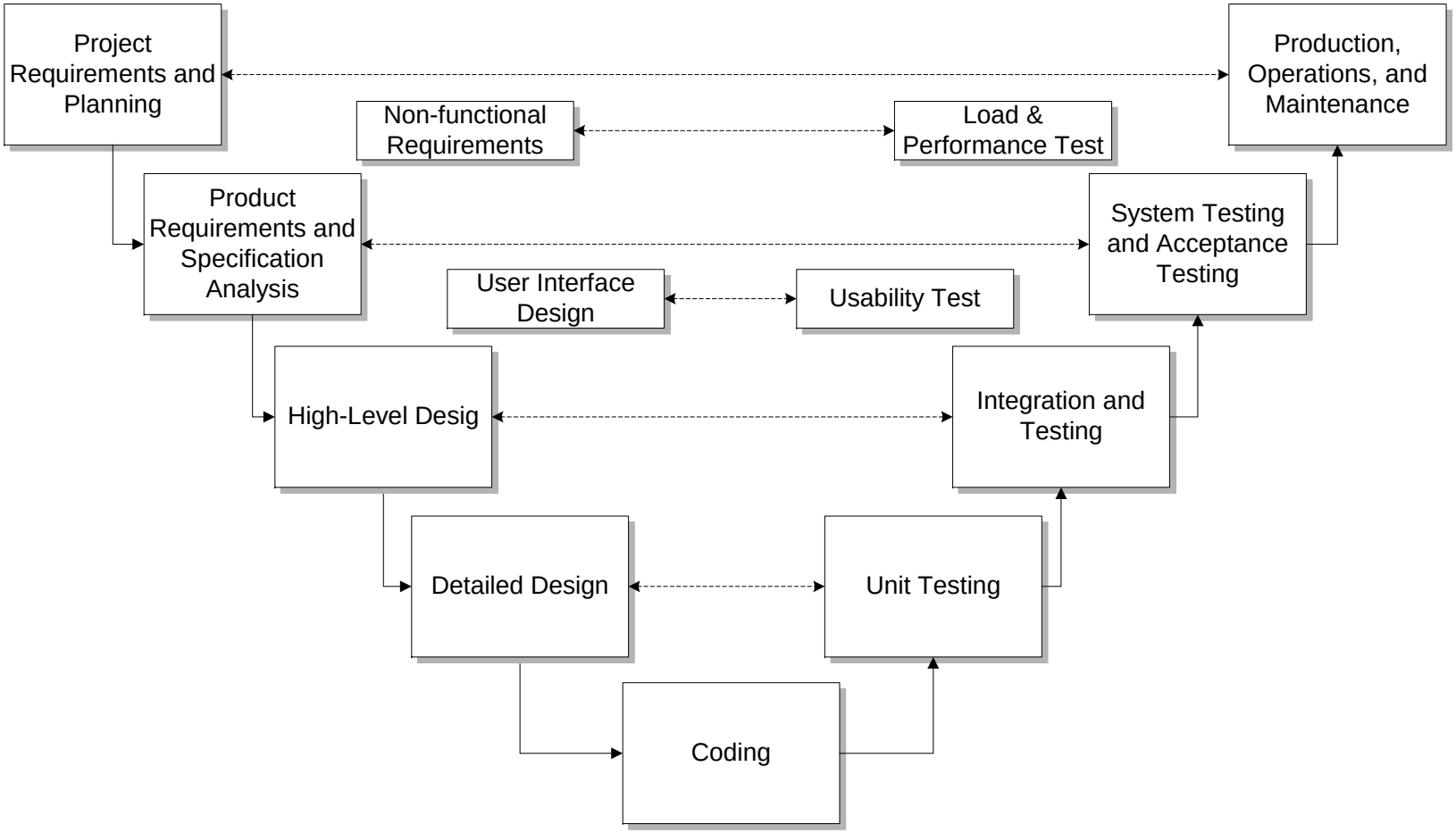
... takže:

Def.: Test je úspěšný, pokud neodhalí žádné anomálie na výstupu programu.

Def.: Test je úspěšný, pokud zjistí přítomnost jedné či více chyb v programu.

- Myers, 1979

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

Úplné testy

- I u malých programů může být počet různých logických cest ohromný.
- Program se 100 řádky, několik vnořených cyklů, každý proveden 20 krát. Existuje přibližně 10^{14} možných cest, které mohou být provedeny.
- Při rychlosti 1 test/ms by testování trvalo 3170 roků!
- Úplné testování není realizovatelné.

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

„Černé a bílé skříňky“

FUNKCE
- test činnosti každé funkce
- test ČERNÉ SKŘÍŇKY

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

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)

Návrh testu „černé skříňky“ - příklad

s := Hledej (NejakePole,UlozenaHodnota)

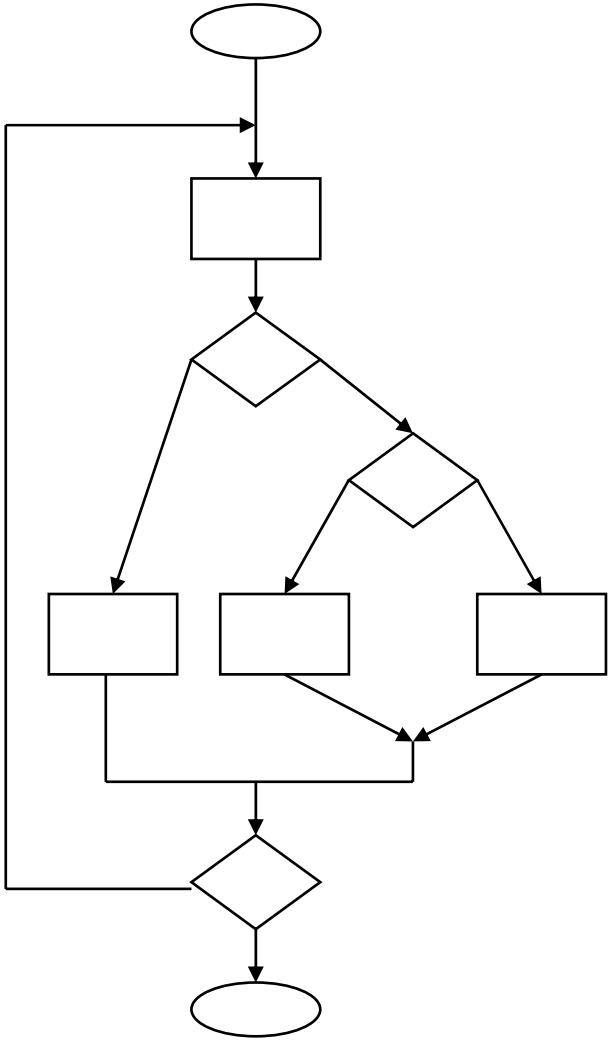
	velikost pole	*hledaný prvek*
1.	1	existuje
2.	1	není
3.	0	
4.	sudé	je první
5.	sudé	je poslední
6.	sudé	není
7.	liché	je první
8.	liché	je poslední
9.	liché	není
10.	sudé	je v obecné pozici
11.	liché	je v obecné pozici

V tomto testu je obsažena zkušenost s mnoha verzemi vyhledávacích programů.

Testování „bílá skříňka“

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

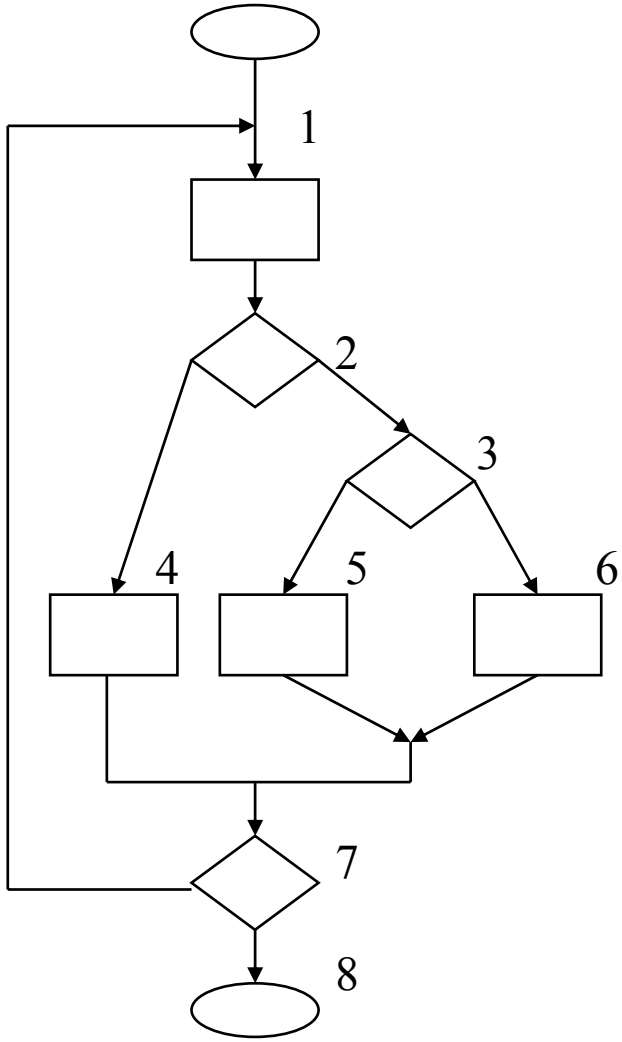
Testování „bílá skříňka“



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 „bílá skříňka“

- Vývojový diagram není nutný, ale obrázek pomůže vysledovat příslušné cesty.
- Testy základních cest by měly být provedeny u kritických modulů.

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

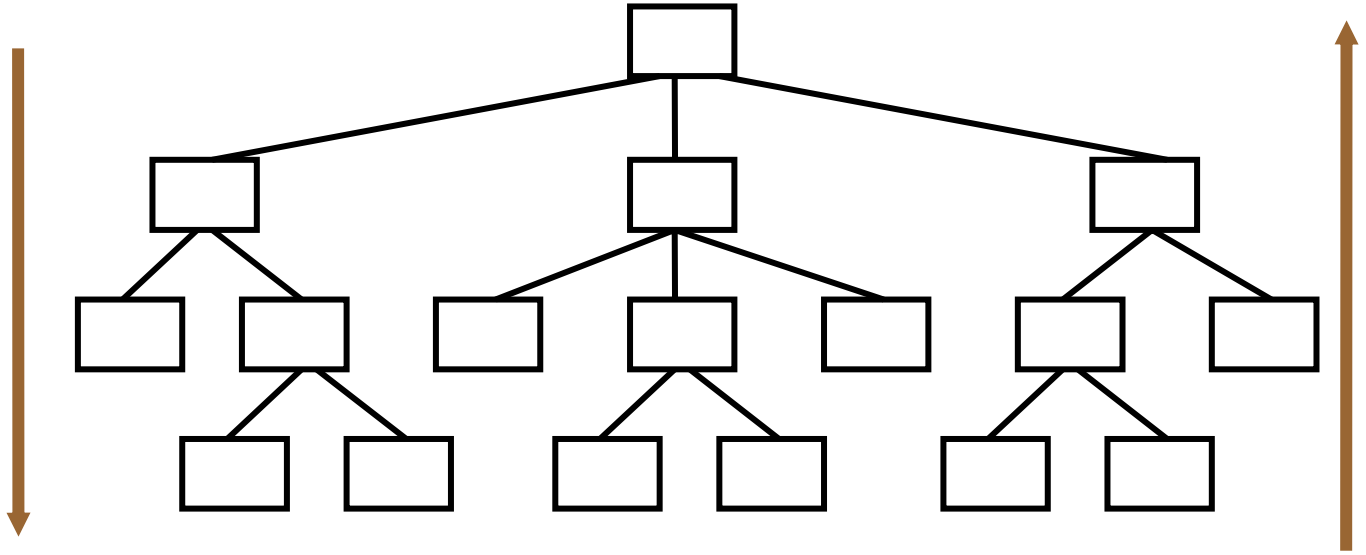
Shora dolů

- 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.

Nevýhody TDT a BUT

Nevýhody TDT:

- Složité objekty, moduly, nelze jednoduše zaměnit za „protézu“.
- Výsledky testů na vyšších úrovních nemusí bý přímo „viditelné“.

Nevýhody BUT:

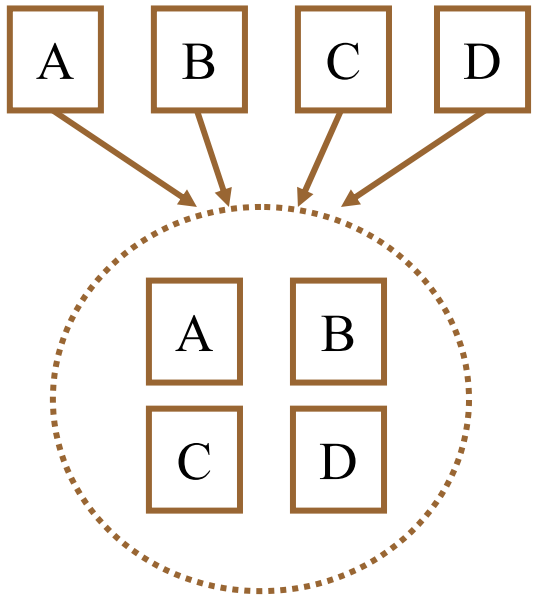
- Čas a náklady na konstrukci „drivers“ pro testování jsou obvykle vyšší, než u „protéz“.
- Až v závěru vznikne program použitelný pro předvedení, ve formě „prototypu“.

Obě metody mají své nevýhody, nelze říci, že jedna je nejlepší.

Atributy integrace

- Kdo dělá integrační testování?
  - vývojářský a/nebo QA tým
- Počet pracovníků a rozpočet jsou na vrcholu
- „Jde do tuhého“
- Problémy:
  - práce pod tlakem
  - blíží se datum odevzdání
  - neočekávaná selhání (vši)
  - motivační problémy
  - konflikty při přejímání zákazníkem

Integrační testování



Testování modulů

Integrační testování

Kde je chyba ?

⇩

Inkrementální integrace a testování

Úkoly

- Pro svůj projekt (část projektu) vytvořte plán testování a začleňte jej do celkového plánu projektu (pokud jste tak již neučinili).
- Pro projekt navrhněte příklad konkrétního testu typu „černá skříňka“.