12 Zajištění kvality programového vybavení

Obecně *dva druhy kvality* u technických produktů:

a) ***Kvalita návrhu* - vlastnosti komponent, specifikované návrháři. U SW se týká analýzy a specifikace požadavků a návrhu.**

b) ***Kvalita shody* - stupeň souladu výroby se specifikací návrhu. U SW se týká implementace.**

***Řízení kvality* - série inspekcí, posuzování a testů prováděných během vývoje s cílem zajistit, že každý výsledek splňuje požadavky na něj kladené.**

***Zajištění kvality* - funkce spojené s kontrolami a zprávami pro management.**

***Náklady kvality*:**

a) prevence - plán kvality, posuzování, testovací podpora, školení, …

b) odstranění chyb - vnitřní (před odevzdáním) - ladění, odstranění, …

- vnější (po předání) - řešení reklamace, záruční práce, zaslání opravené verze, hot line, …

- náklady na odstranění rostou v závislosti na fázi vývoje.

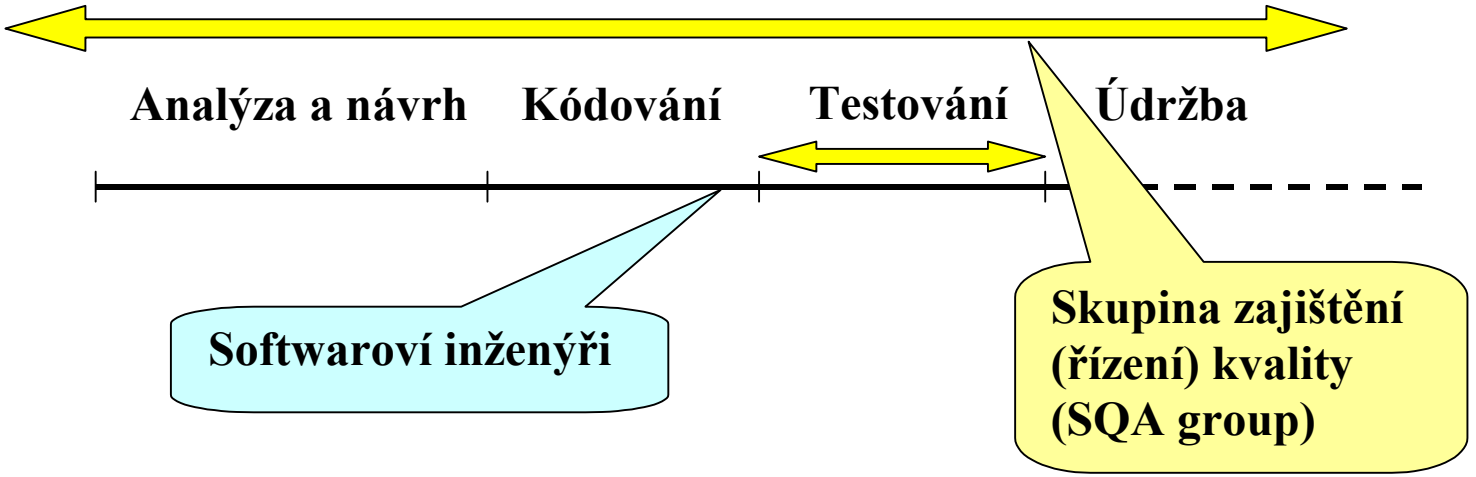
12.1 Kvalita programového vybavení

Kvalita programového vybavení je soulad s explicitně stanovenými funkčními a výkonnostními požadavky, explicitně dokumentovanými vývojovými standardy a implicitními vlastnostmi, které jsou očekávány od každého profesionálně vyvíjeného programového vybavení.

Faktory ovlivňující kvalitu (Mc Call):

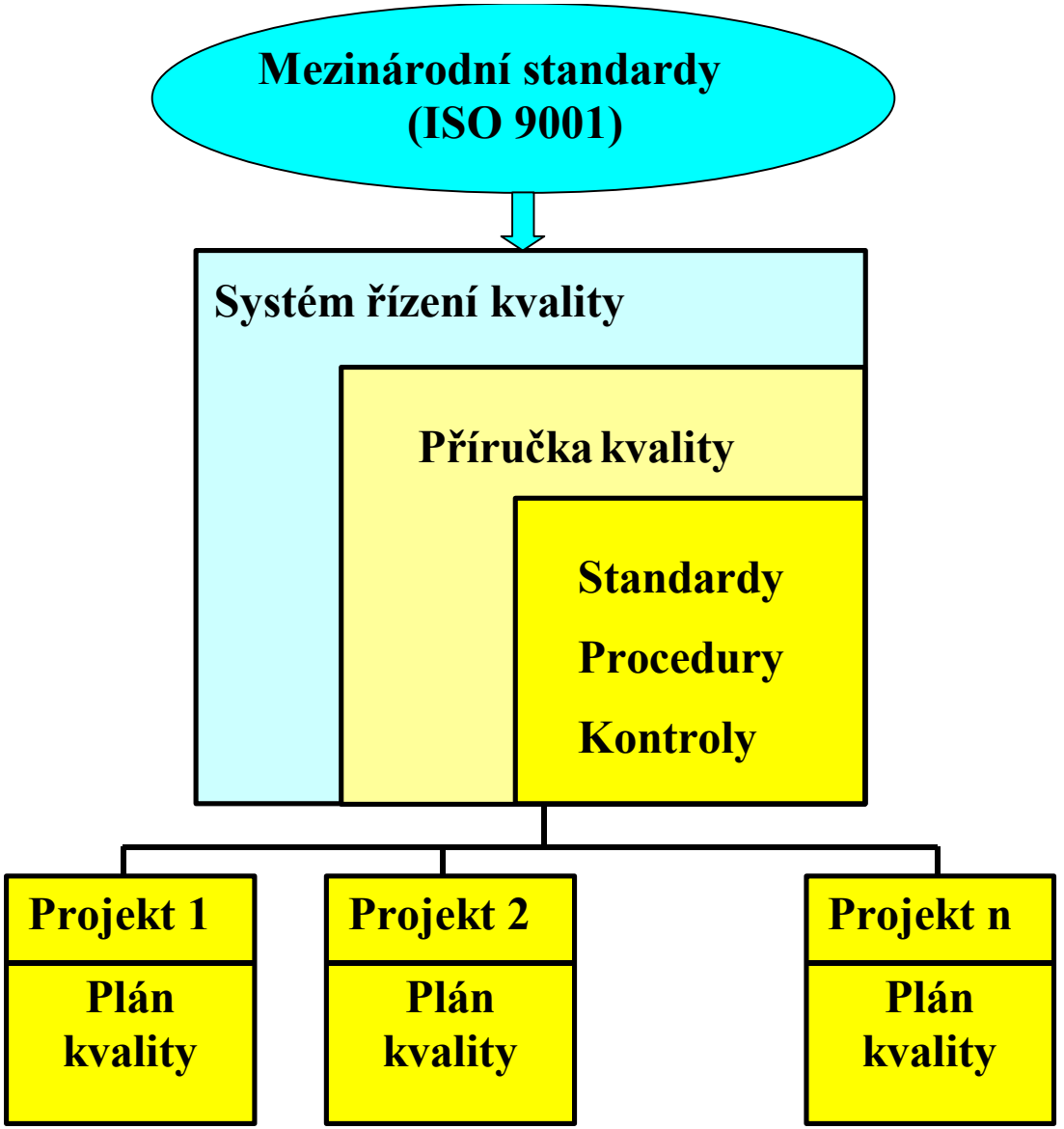
a) **Operace produktu (správnost, spolehlivost, efektivita, integrita, použitelnost)**
b) **Revize produktu (udržovatelnost, flexibilita, testovatelnost)**
c) **Přechod produktu (přenositelnost, opakované použití, interoperabilita)**

***Zajištění kvality SW* - plánovaný systematický sled akcí, vyžadovaný k zajištění kvality.**



Aktivity skupiny zajištění kvality:

- **příprava plánu kvality projektu (prováděná hodnocení, audity a oponování, standardy použité v projektu, dokumentace, plán testů, postupy při zjištění chyb)**
- **účast na tvorbě popisu procesu projektu (inženýři vyberou, skupina posuzuje)**
- **oponování vývojových aktivit a výsledků aktivit**
- **dokumentování odchylek od stanoveného procesu a tvorba zpráv pro management**
- **řízení a správa změn, sběr a analýza softwarových metrik, …**


Mezinárodní standardy
(ISO 9001)
Systém řízení kvality
Příručka kvality
Standardy
Procedury
Kontroly
Projekt 1
Plán kvality
Projekt 2
Plán kvality
Projekt n
Plán kvality

Př) Doporučení pro zápis programů v C++

Převzato z knihy Maths Henricson, Erik Nyquist: Industrial Strength C++. Prentice Hall, 1997, ISBN 0-13-120965-5

♦ **Pojmenování**

Smysluplná jména

◊ **Používejte smysluplná jména.**

◊ **Pro identifikátory používejte anglická jména**

◊ **Buďte konzistentní při pojmenovávání funkcí, typů, proměnných a konstant.**

Kolidující jména

◊ **Globální by měla být pouze jména pro `namespace`.**

◊ **Nepoužívejte globální deklarace `using` a direktivy `using` uvnitř vkládaných souborů.**

Nevhodná jména

◊ **Nepoužívejte identifikátory obsahující dvě nebo více podtržení za sebou.**

◊ **Nepoužívejte identifikátory začínající podtržením.**

♦ **Organizace zdrojového kódu**

◊ **Každý vkládaný soubor by měl být soběstačný.**

◊ **Vyhýbejte se vkládání nepotřebných souborů.**

◊ **Zabraňte vícenásobnému vkládání uzavřením veškerého kódu ve vkládaných souborech do podmíněného překladu.**

◊ **Definice inline metod by měly být umístěny v samostatném souboru. …**

12.2 Formální oponentury (posuzování)

- „filtr“ procesu vývoje, „vyčištění“ produktů vývoje

Důvody: 1. Mýlit se je lidské.
2. Autor přehlédne řadu chyb.

Cíl: odhalení chyb, upozornění na potřebná vylepšení, sjednocení způsobu vývoje

Př.) Walkthrough
- schůzky: 3 až 5 lidí, příprava účastníků (max. 2 hod.), trvání max. 2 hod.
- účastníci: vedoucí oponent, oponenti, autor
- procházení $\rightarrow$ závěr (přijetí/nutnost menších modifikací/odmítnutí)

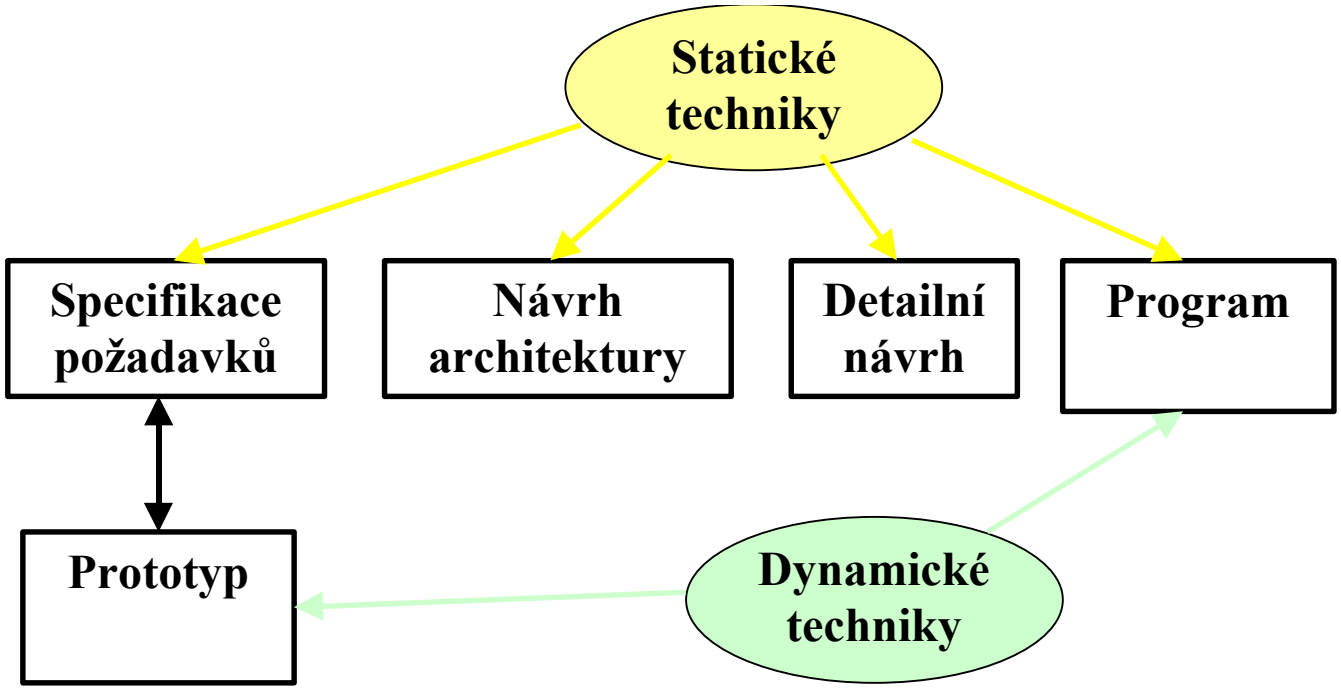
12.3 Verifikace a validace

Validace: „Vytváříme správný produkt?“

Verifikace: „Vytváříme produkt správně?“

Techniky: a) statické

b) dynamické (testování)



Testování: a) testování chyb (defect testing)

b) statistické testování - výkonnostní a spolehlivostní

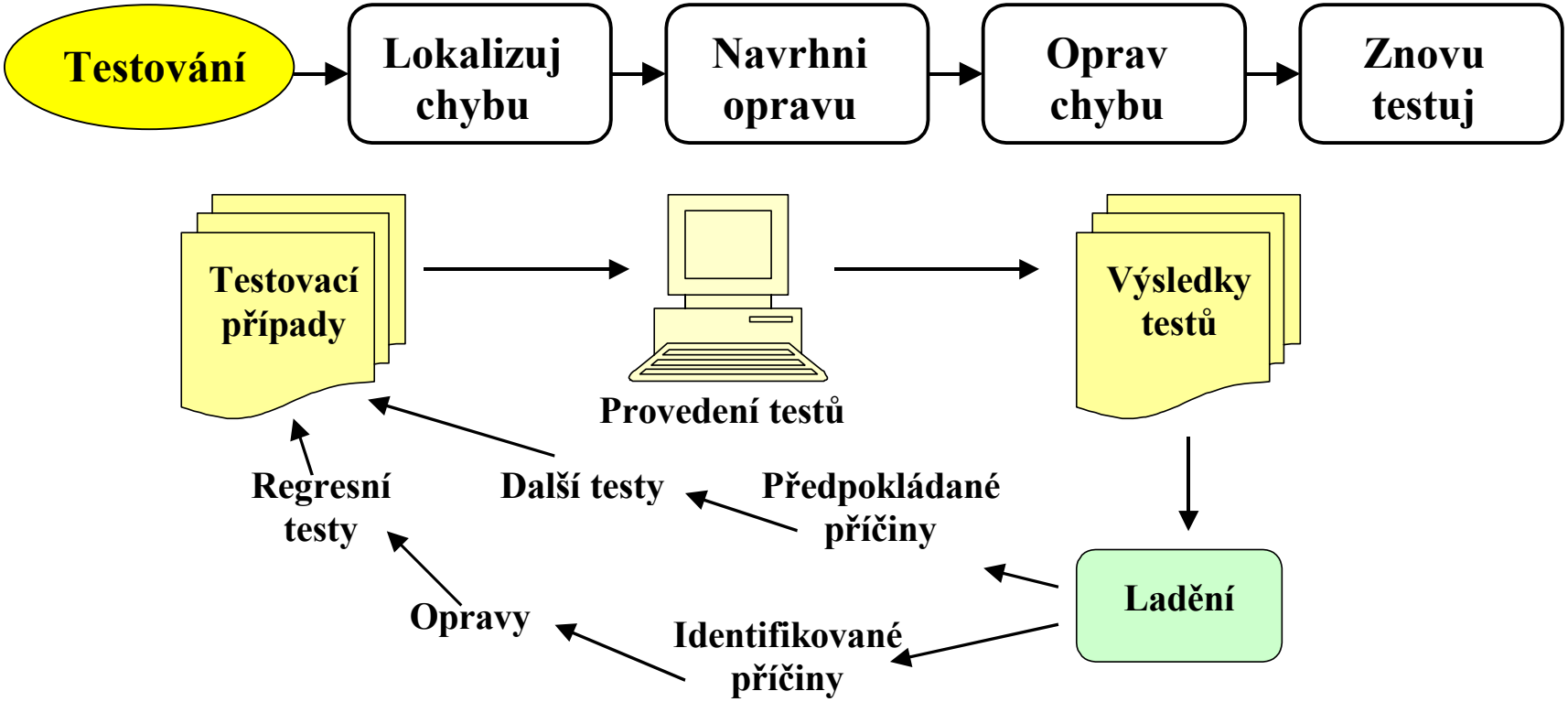
- **Testování chyb**

Dobrý test **- test s vysokou pravděpodobností odhalení dosud neodhalených chyb.**

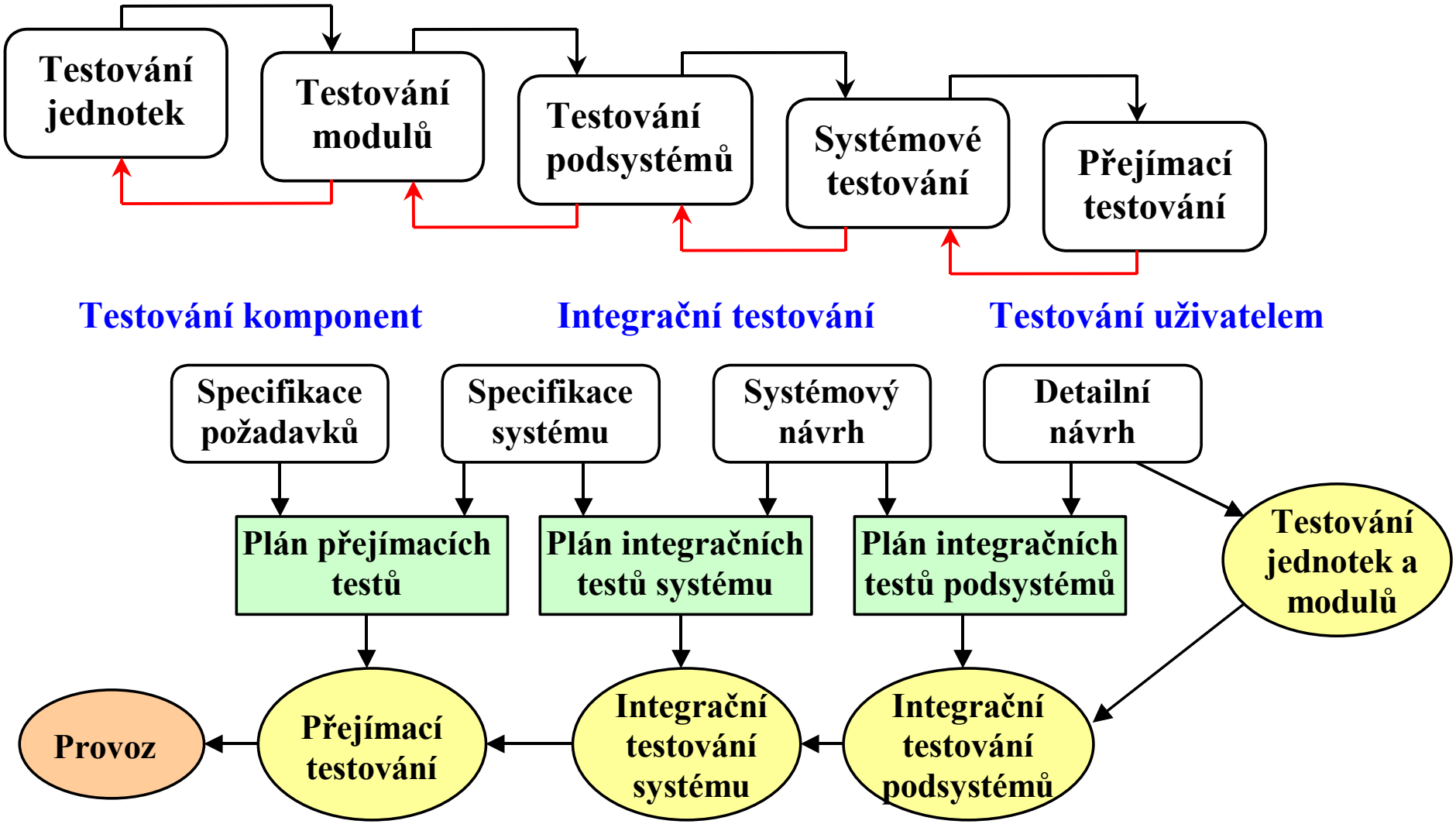
„Úspěšný“ test **- objeví dosud neodhalenou chybu.**

Testování nemůže prokázat nepřítomnost chyb, může pouze ukázat, že v programu chyby jsou.

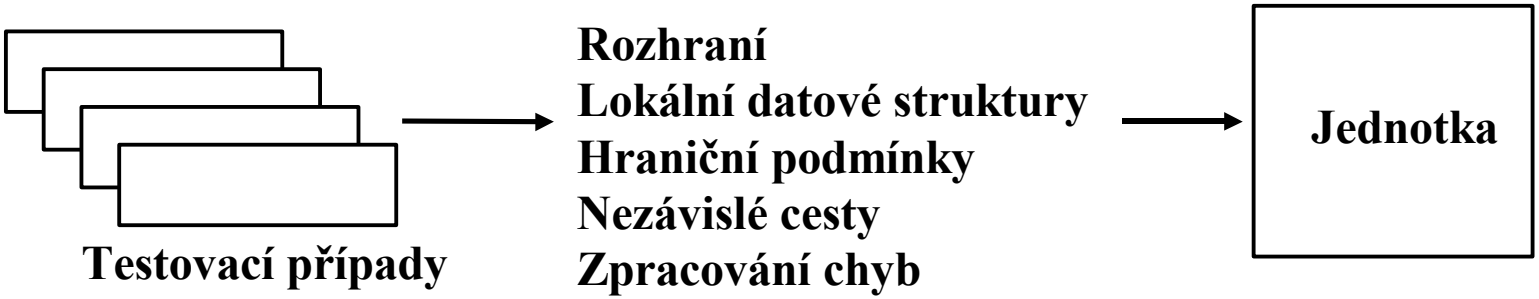
- **Ladění**

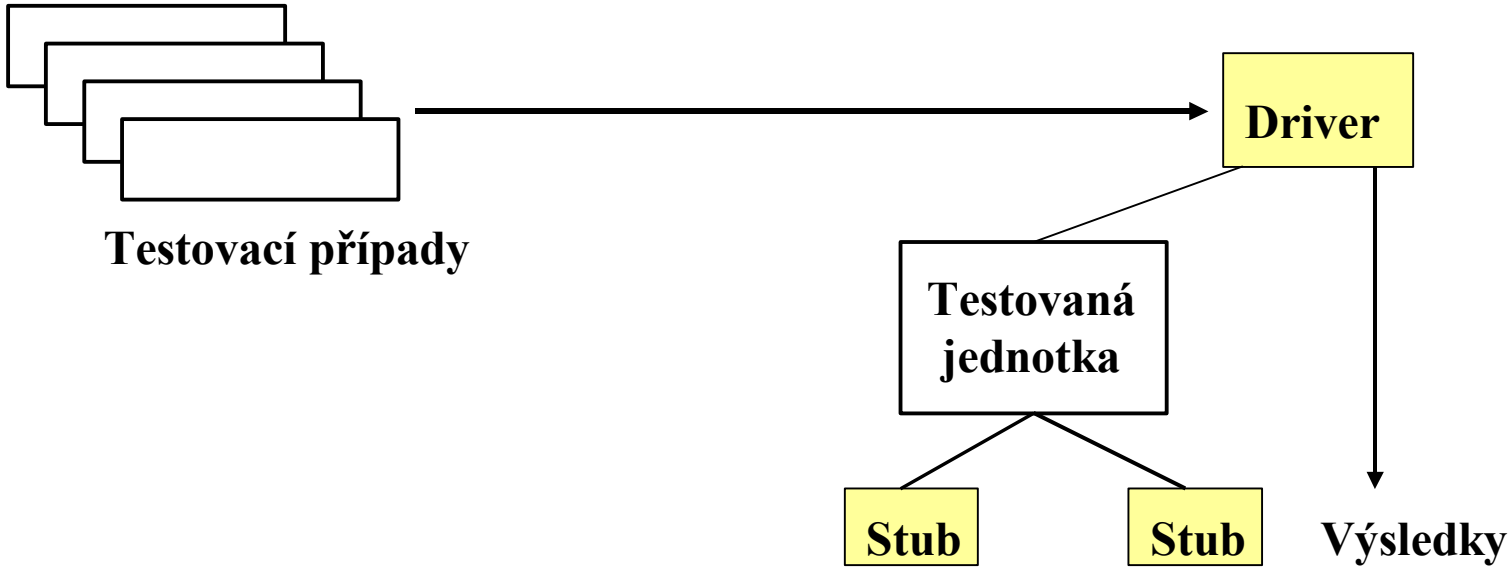


12.4 Strategie testování



- **Testování jednotek**



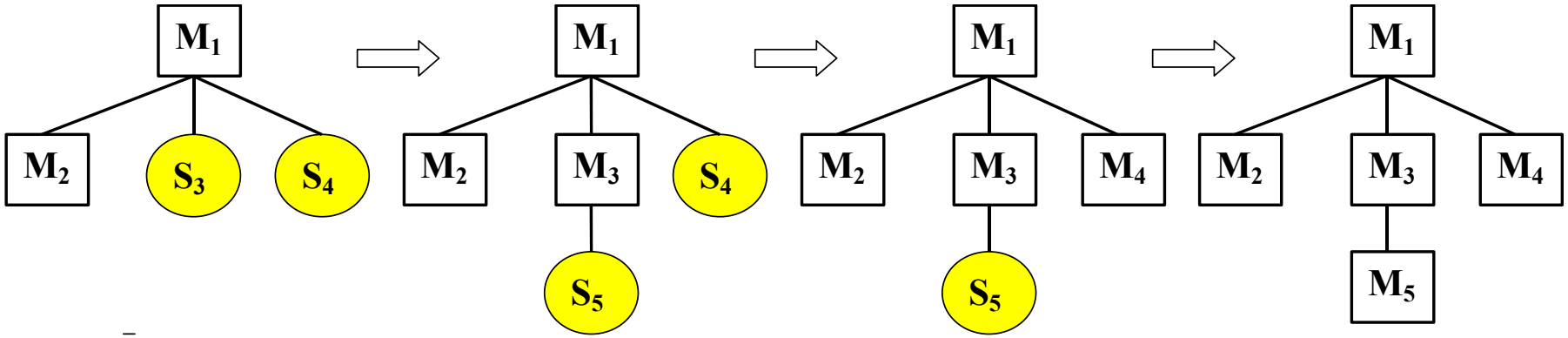


- **Integrační testování**

a) neinkrementální

b) inkrementální

shora - dolů



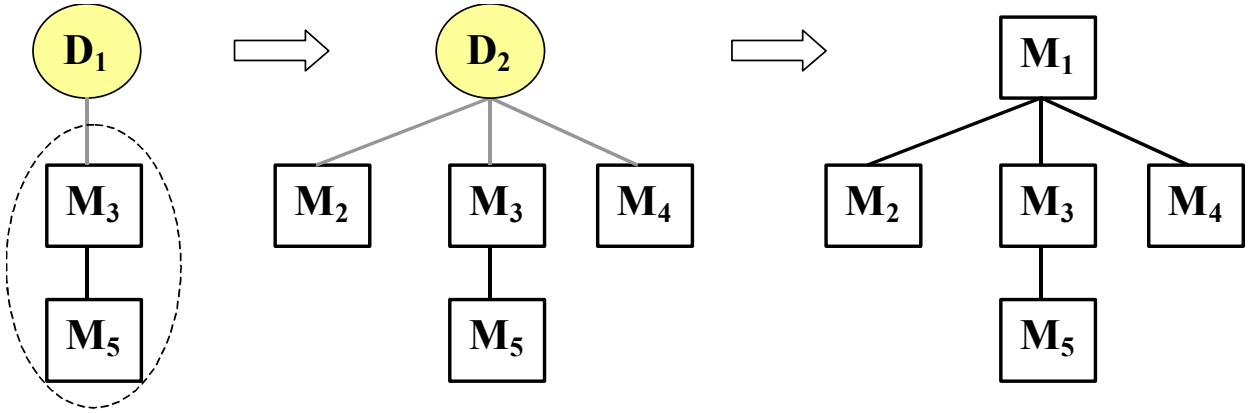
- měl by být použit při vývoji programu shora dolů, evolučním prototypování apod.

- příklad „náhražek“: zobrazení zprávy, zobrazení předaného parametru, vrácení hodnoty z tabulky, souboru nebo zadané

+ možnost odhalení chyb (strukturních) v ranném stádiu programování, brzy je k dispozici systém s omezenou funkčností (psychologický efekt, možnost validace)

- nutnost implementace „náhražek“

zdola - nahoru



- příklad „driveru“: spuštění podřízeného, zobrazení vráceného parametru, předání hodnoty z tabulky, souboru nebo zadané
- použití zpravidla pro „kritické komponenty“, jinak shora dolů

Regresní testování

- po přidání modulu, resp. provedené změně (údržba).

Cíl: Ověření, že to, co fungovalo před přidáním modulu (změnou), funguje i po něm porovnáním výsledků testů.

Zahrnuje:

- reprezentativní vzorek testů ověřujících všechny funkce,
- nové testy, zaměřující se na funkce, které pravděpodobně byly ovlivněny,
- testy zaměřující se na nové funkce, resp. změny.

- problém regresního testování interakčních programů s GUI.

- **Validační testování**

Validace je úspěšná, funguje-li programové vybavení tak, jak to zřejmě očekává zákazník.

Dva případy:

a) programové vybavení vytvářeno pro jednoho zákazníka
→ přejímací testy (alfa testování)

b) programové vybavení vytvořeno jako produkt na trh
→ beta testování

- **Systémové testování (kompletní systém s počítači)**

- zaměření se na vlastnosti, které nelze ověřit samostatně (výkonnost, bezpečnost, zotavení, …)

- **Některé speciální typy testů**

  ***Výkonnostní, kapacitní* – vliv zátěže.**

  ***Operační* – delší dobu v provozních podmínkách**

  ***V plném provozu (full scale)* – pokračování operačních, přiblížení se limitu**

  ***Přetížení (stress)* – chyby při přetížení, chování**

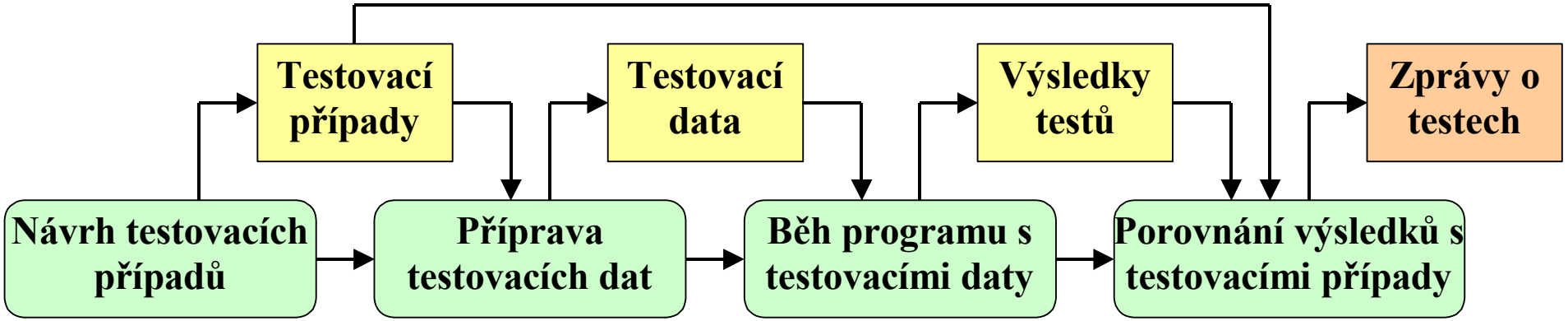
  ***Negativní* – chybné používání.**

  ***Ergonomické* – uživatelské rozhraní (konzistence, čitelnost, barvy, nápověda, …)**

  ***Dokumentace* – instalace, použití (srozumitelnost, aktuálnost, vyváženost kapitol,…**

  ***Regresní* – viz integrační testování.**

12.5 Techniky testování chyb



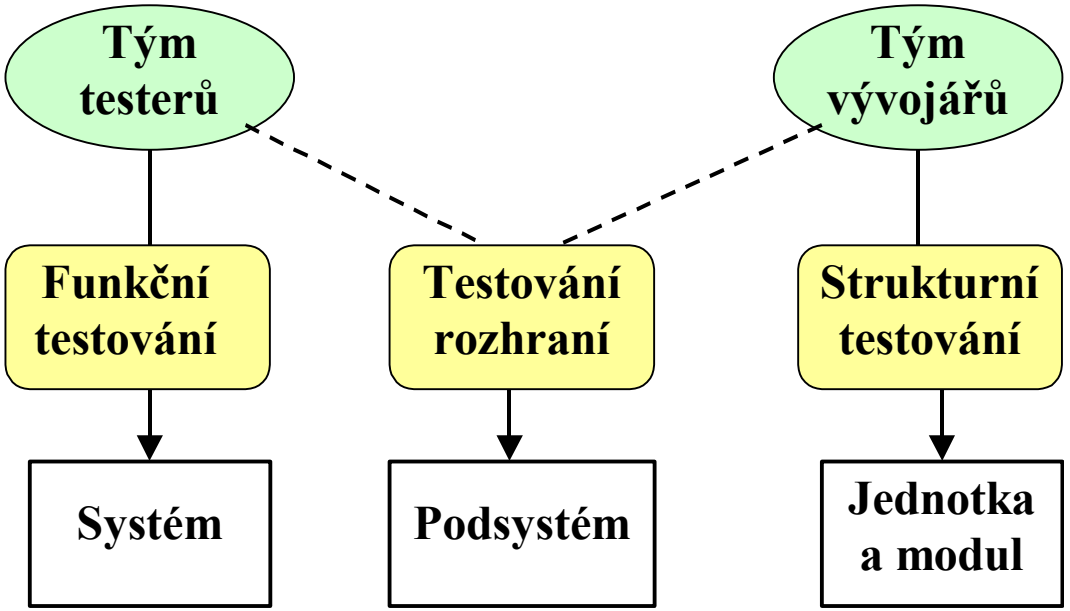
- úplné testování prakticky nemožné → podmnožina případů, např.:
  - každý příkaz programu proveden alespoň jednou nebo
  - testování schopností systému důležitější než komponent, testování starých schopností (u nového systému) důležitější než nových, testování typických situací mnohem důležitější než okrajových

Tři přístupy k testování:

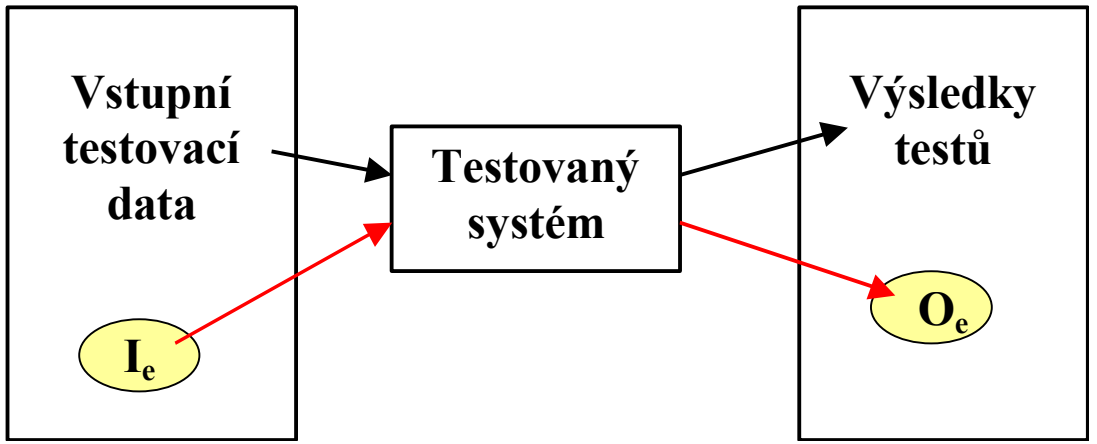
a) Funkční (black-box),

b) Strukturní (white-box),

c) Testování rozhraní



- **Funkční testování**



Problém: nalezení co nejvíce dat z I_e

Metoda rozčlenění podle ekvivalence (equivalence partitioning)

- rozčlenění dat do tříd ekvivalence a výběr kandidátů

Př) Test podprogramu pro vyhledání prvku v poli

```
procedure Search (Key: ELEM; T: ELEM_ARRAY;
                  Found: out BOOLEAN; L: out ELEM_INDEX);
Pre-condition
    TFIRST <= TLAST -- the array has at least one element
Post-condition
-- the element is found and is referenced by L
    (Found and T(L) = Key)
or
-- the element is not in the array
    (not Found and not (exists i, TFIRST>= i <= TLAST, T(i) = Key))
```

Třídy ekvivalence:

a) prvek je-není v poli

b) pole má jeden-více prvků

c) hledaný prvek je první-poslední-uprostřed

- testování hraničních podmínek a výjimečných situací

- **Strukturní testování**

- tester zná kód a může využít znalosti struktury testované části

Testování cest

- prověření všech nezávislých cest komponenty
(→ každý příkaz proveden nejméně jednou, každé větvení testováno pro obě podmínky)

- počet nezávislých cest lze zjistit analýzou grafu toku řízení

Testování podmínek

- testování všech logických podmínek (chyby operátorů, precedence, výrazů v podmínkách, …)

Testování cyklů

- testování konstrukcí cyklů

- **Testování rozhraní**

- odhalení chyb v důsledku chyb rozhraní nebo chybných předpokladů

- důležité u OO vývoje (opakované použití, interakce)

Typy rozhraní:
a) parametry
b) sdílená paměť

c) procedurální (objekty, ADT)
d) předávání zpráv

Typy chyb:

a) chybné použití rozhraní (typ, počet, pořadí parametrů)

b) chybné pochopení specifikace a rozhraní

c) chyby v časování (hlavně RT systémy)

- význam typových kontrol a inspekcí programu.