TS

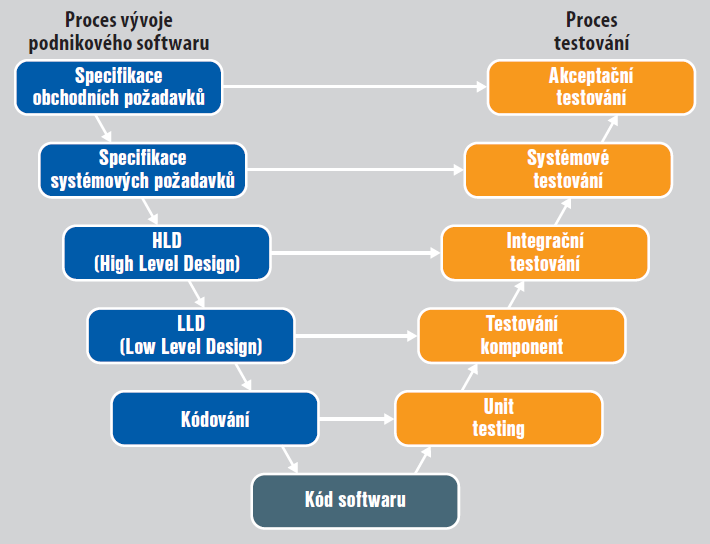
Techniky návrhu testovacích scénářů, jednotkové testy, automatické funkční testy, testovací strategie, testovací data (TS1)

# Pojmy

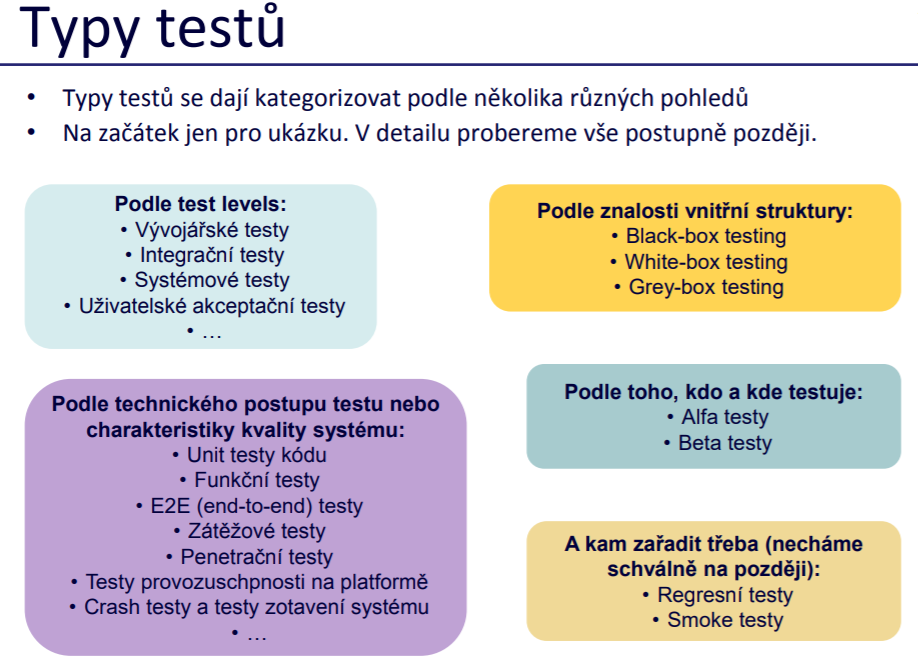
* **Severita**: závažnost chyby z hlediska systému
* **Priorita**: důležitost opravy z hlediska programátora
* **Verifikace:** proces vyhodnocování, jestli produkt konkrétní vývojové fáze odpovídá požadavkům na tento produkt definovaným na začátku této fáze.
* **Validace:** vyhodnocování software v průběhu nebo na konci jeho vývoje, jestli software odpovídá zadaným požadavkům.
* **Regrese:** zanášení chyb při odstraňování jiných.

# Zásady testování

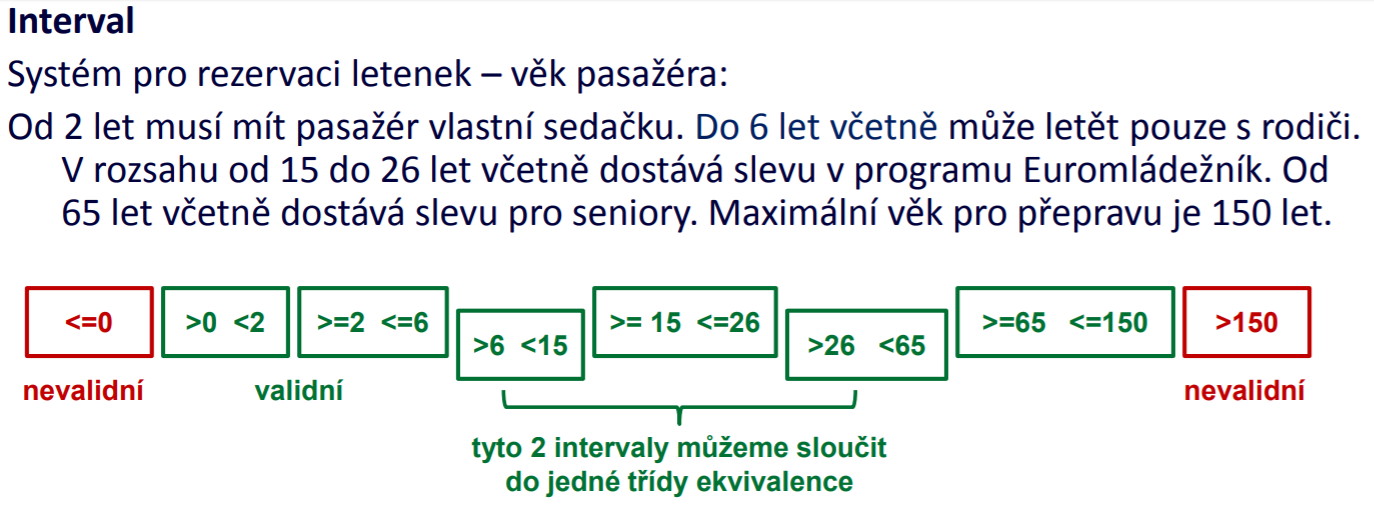
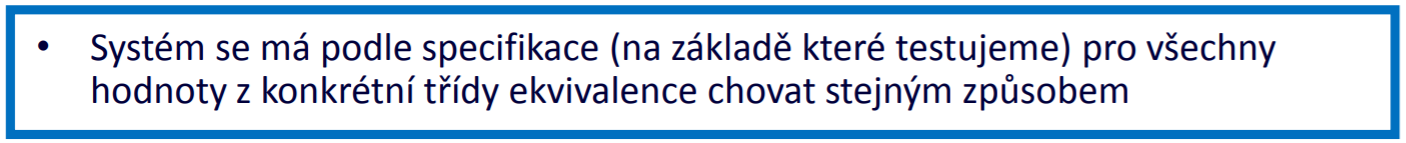
* **Testování ukazuje přítomnost defektů**
  + Testování může ukázat, že jsou defekty přítomny, ale nemůže dokázat, že v softwaru nejsou žádné defekty.
  + Testování snižuje pravděpodobnost, že v softwaru zůstanou neobjevené defekty, avšak nenalezení žádného defektu stále není důkaz bezchybnosti.
* **Vyčerpávající testování je nemožné**
  + Testování všeho (všech kombinací vstupů a vstupních podmínek) není realizovatelné s výjimkou triviálních případů.
  + Namísto vyčerpávajícího testování by měly být k určení hlavního předmětu testovacího úsilí použity analýza rizik a stanovení priorit.
* **Včasné testování**
  + Testovací aktivity by měly v rámci životního cyklu vývoje softwaru nebo systému začít tak brzy, jak je to jen možné, a měly by být zaměřeny na definované cíle.
* **Shlukování defektů**
  + Malé množství modulů obsahuje většinu defektů zjištěných v průběhu testování před uvolněním, nebo je zodpovědné za nejvíc provozních selhání.
* **Pesticidní paradox** 
  + Jsou-li stále opakovány tytéž testy, časem stejný soubor testovacích případů nenalezne žádné nové defekty.
  + K překonání tohoto „pesticidního paradoxu“ je potřeba existující testovací případy pravidelně revidovat a upravovat a je potřeba napsat nové, odlišné testy na vykonání jiných častí softwaru nebo systému pro případné odhalení dalších defektů.
* **Testování je závislé na kontextu**
  + Testování je vykonáváno odlišně v různých kontextech.
  + Například software kritický z pohledu bezpečnosti se testuje jiným způsobem než webové stránky elektronického obchodu.
* **Falešná představa o neexistenci omylů**
  + Nalezení a opravení defektů nepomůže, pokud je vytvořený systém nepoužitelný a nesplňuje potřeby a očekávaní uživatelů.



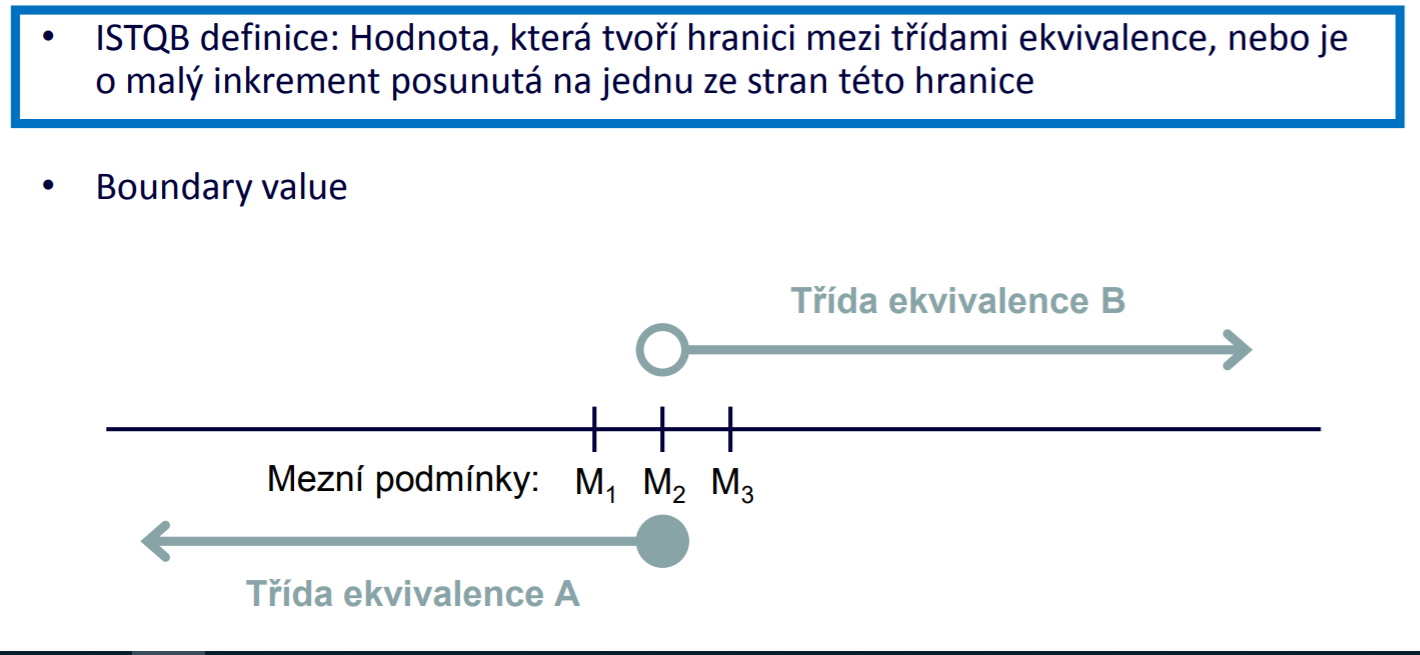
# Typy testů



## **Třídy ekvivalence**



**Mezní podmínky**



## Vývojářské testy (unit testy)

Jde o první fázi testování, která se zaměřuje na nejmenší testovatelné části aplikace. Jde obvykle o testy jednotlivých komponent aplikace na úrovni modulů, objektů a tříd. Tento druh testování obvykle provádějí vývojáři a nebývá zahrnut do plánů testů aplikace. Vývojáři si těmito testy ověřují, že nová nebo změněná část kódu funguje (tedy nepadá do chyby) a že její funkce odpovídá očekávání.

## Integrační testy

Integrační testy obvykle postupují od jednotlivých modulů směrem k věštím celkům. Tedy nejdříve jsou testována rozhraní jednotlivých modulů. Mnohdy jsou v této fázi využívány tzv. fake moduly / mock třídy (potažmo data). Jde o simulátory, jejichž úkolem je napodobovat komunikaci ostatních modulů ale bez jejich funkčnosti. Díky nim se ověří, že modul umí korektně odesílat i přijímat komunikaci s dalšími moduly.

Další fází je spojování modulů do větších celků, které mají z pohledu testování smysl. Poslední fází je testování kompletní aplikace.

## Systémové testy

Pokud se dnes mluví o testování, nejčastěji se tím myslí systémové testy. Jde o ověření, že aplikace jako celek funguje správně. Testuje se, že správně plní úlohu, pro kterou byla vyvinuta, že vrací správné výstupy, že byly ošetřeny všechny nestandardní situace a v neposlední řadě, že byly pokryty všechny požadavky ze strany zákazníka. Systémové testy obvykle probíhají v několika kolech. Jsou hlášeny nalezené chyby, ty jsou opraveny a v následujících kolech retestovány.

## Akceptační testy

Z pohledu dodání aplikace zákazníkovi jsou nejpodstatnější Akceptační testy. Ty mají ověřit, že aplikace splňuje všechny zákazníkovy požadavky. Tyto testy může provádět stejný tým, který provedl systémové testy, ale stejně časté je i to, že zákazník pověří těmito testy jiný tým, který si často pro tyto účel vytvoří. Požadavky, které zákazník na funkci aplikace má, je vhodné ještě před zahájením vývoje (a zcela jistě před zahájením testování) shrnout do tzv. akceptačních kritérií. Ta mohou obsahovat požadavek na maximální počet chyb nalezených při testování, výkonové a zátěžové požadavky a podobně. Splněním těchto požadavků vývoj aplikace v podstatě končí a aplikaci přebírá zákazník.

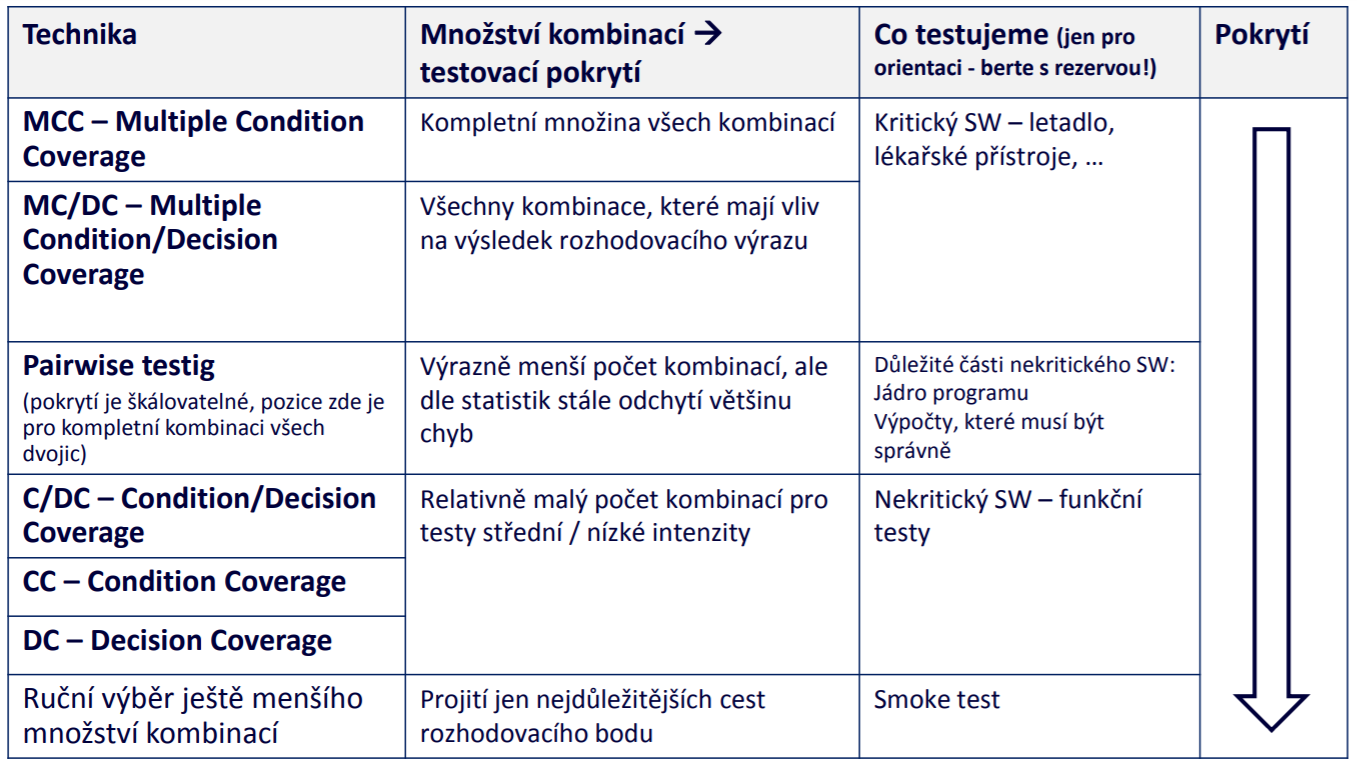
## Zátěžové testy

Především u webových aplikací je potřeba vyzkoušet kolik uživatelů může najednou software obsloužit. Zátěžové testy lze však aplikovat i na desktopové aplikace. Součástí takových testů je i průběžný monitoring serveru a všech aplikací. Z takto získaných dat lze pak vyhodnotit výkonnostní problémy, určit kapacitu aplikace či optimalizovat natavení pro lepší výkon softwaru.

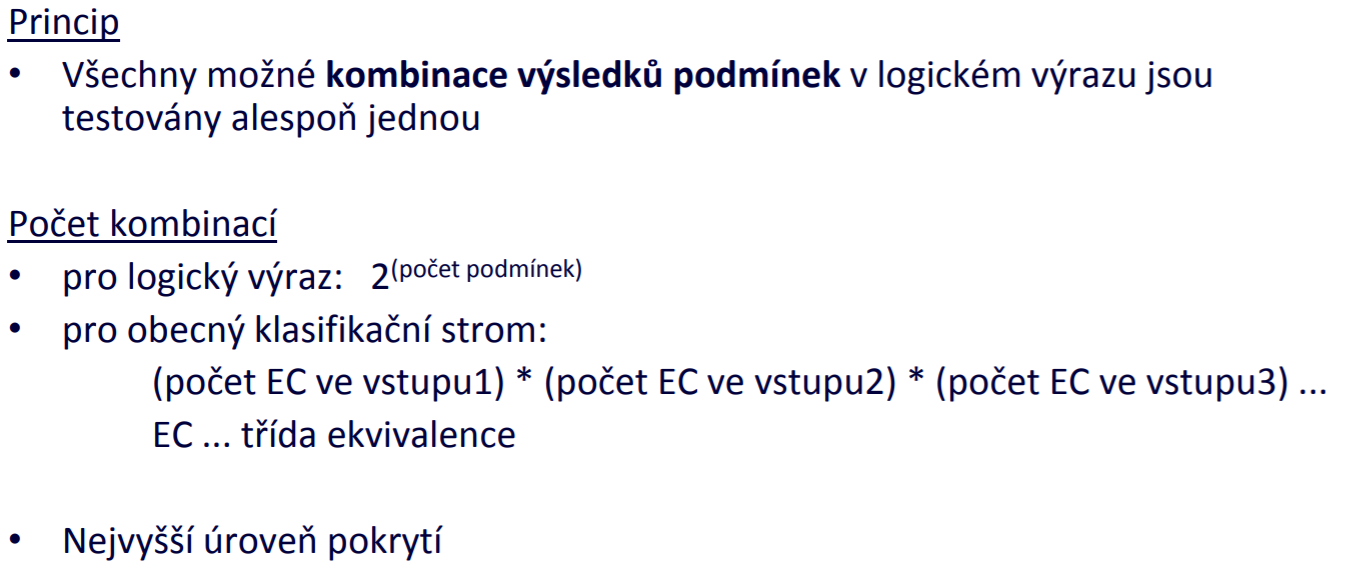
## End-2-End testy

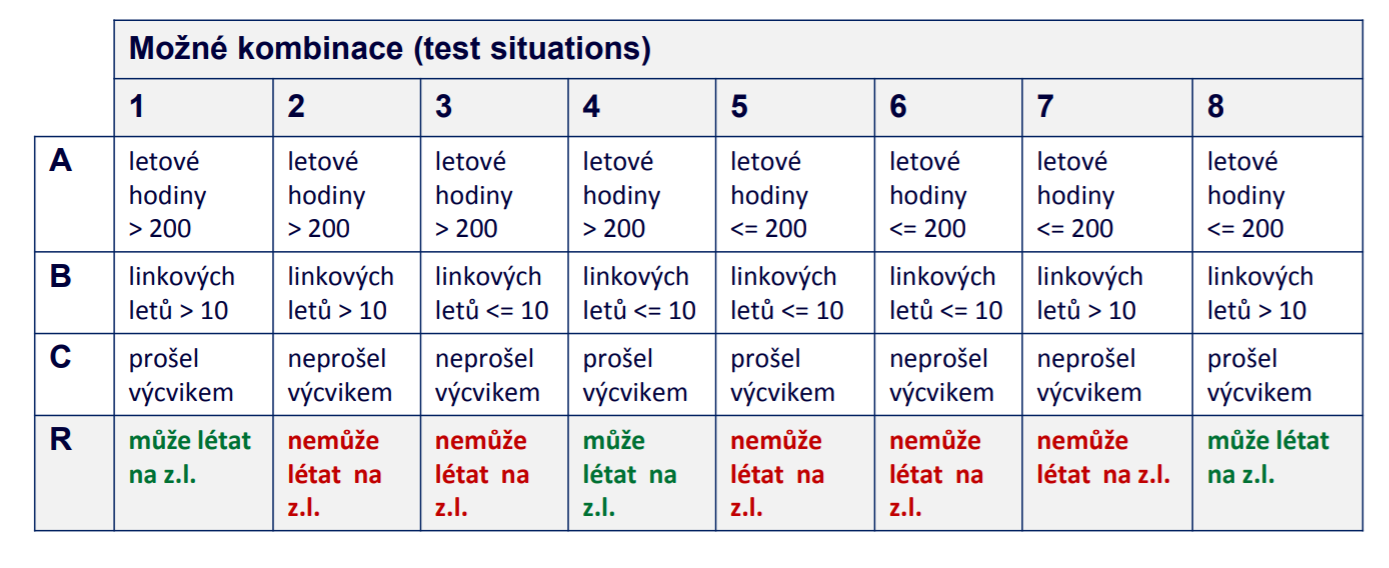
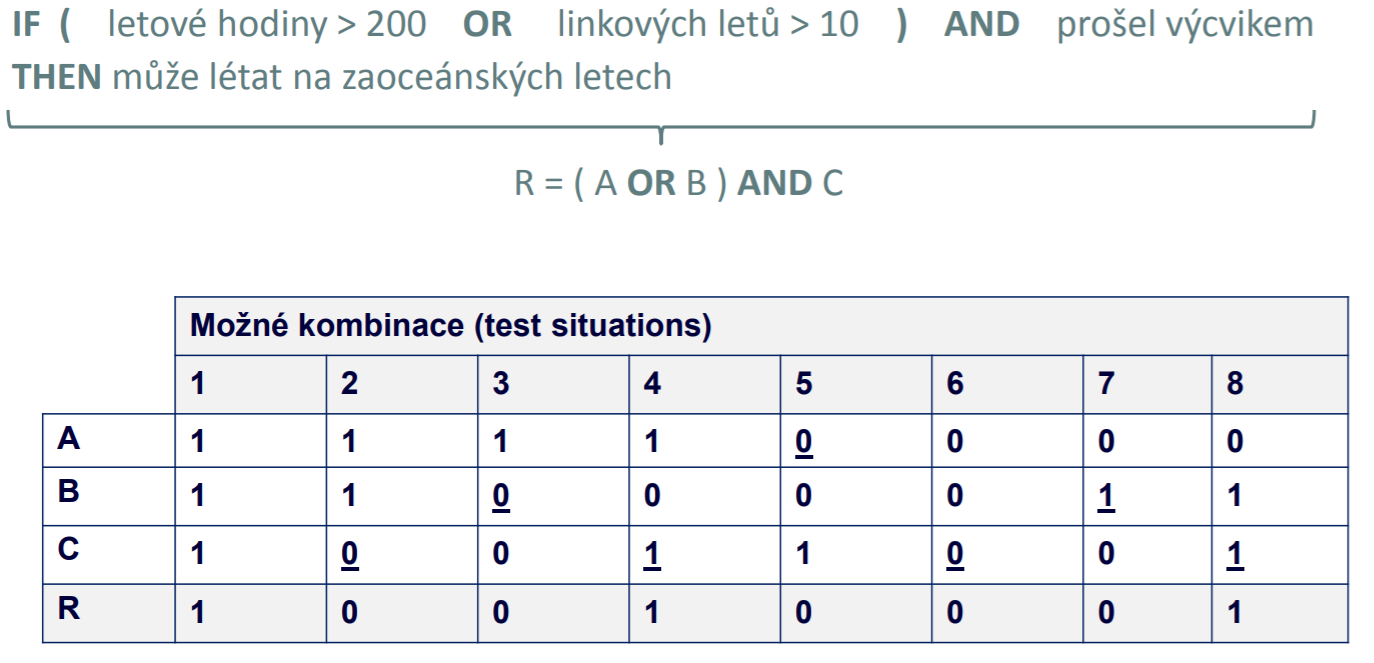
End to end testing je testování aplikace se všemi jejími systémovými rozhraní společně od počátečního bodu po koncový bod pro funkční a datovou integritu. Cílem testování typu end to end je bezpochyby simulovat prostředí podobné realitě se všemi komponenty a rozhraními. Je zřejmé, že e2e testování probíhá po dokončení testování unit testy, integračních a systémových testů.

## Testování – kombinace vstupů

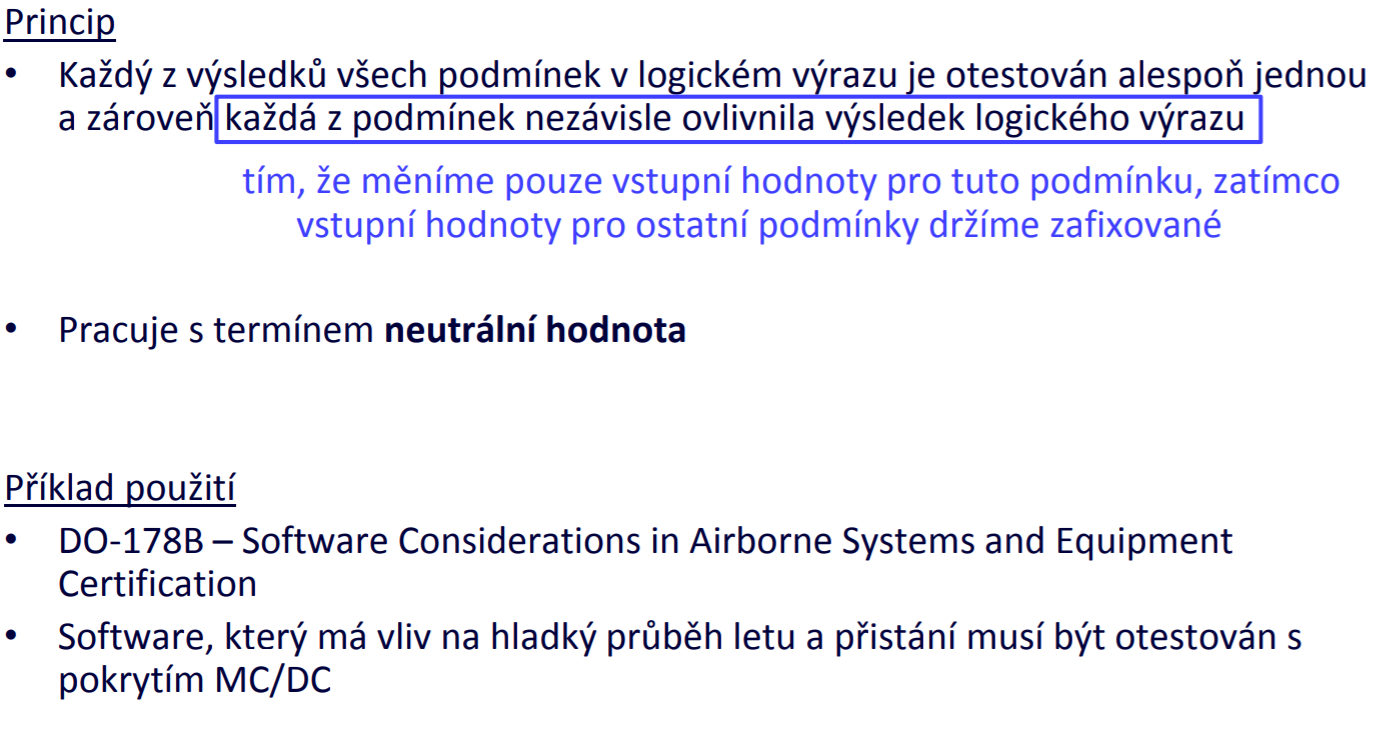


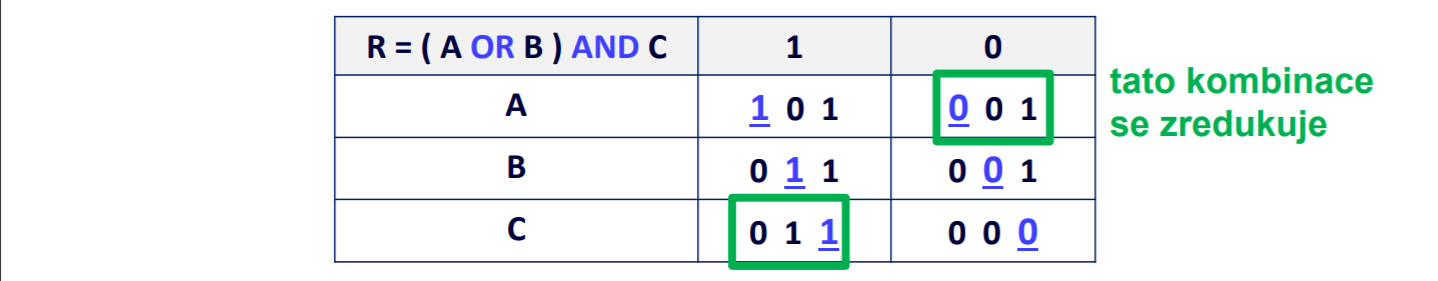
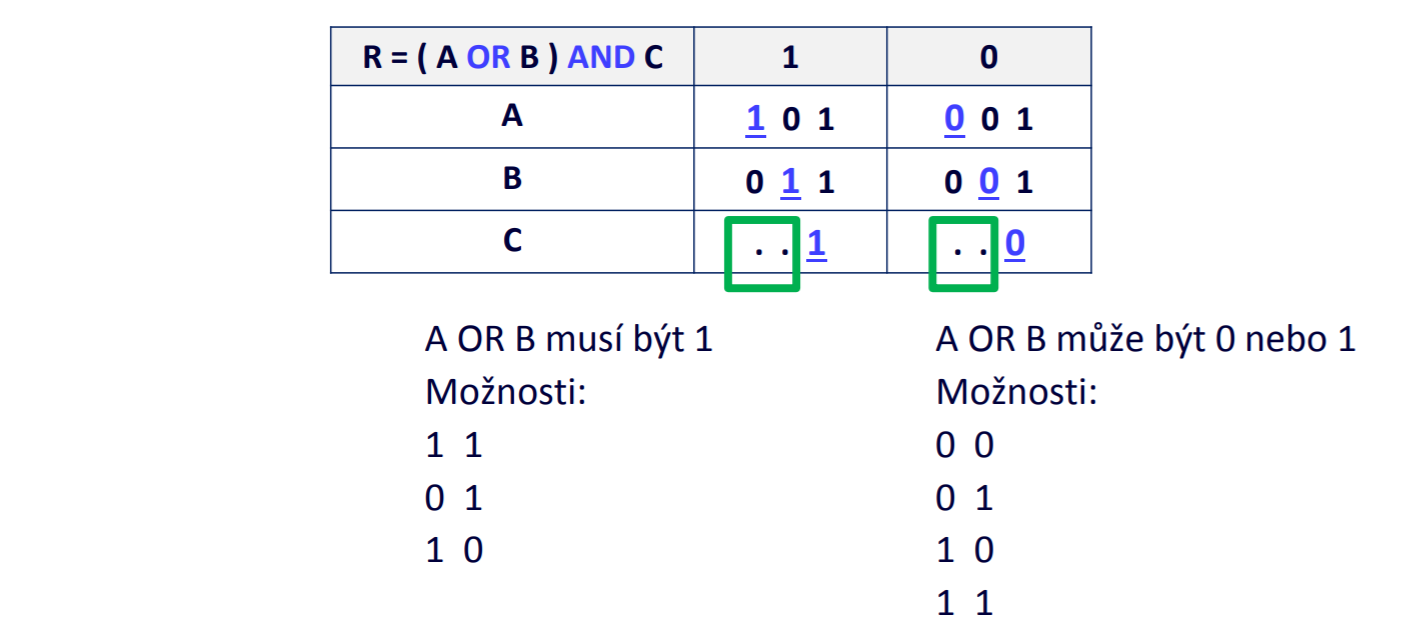
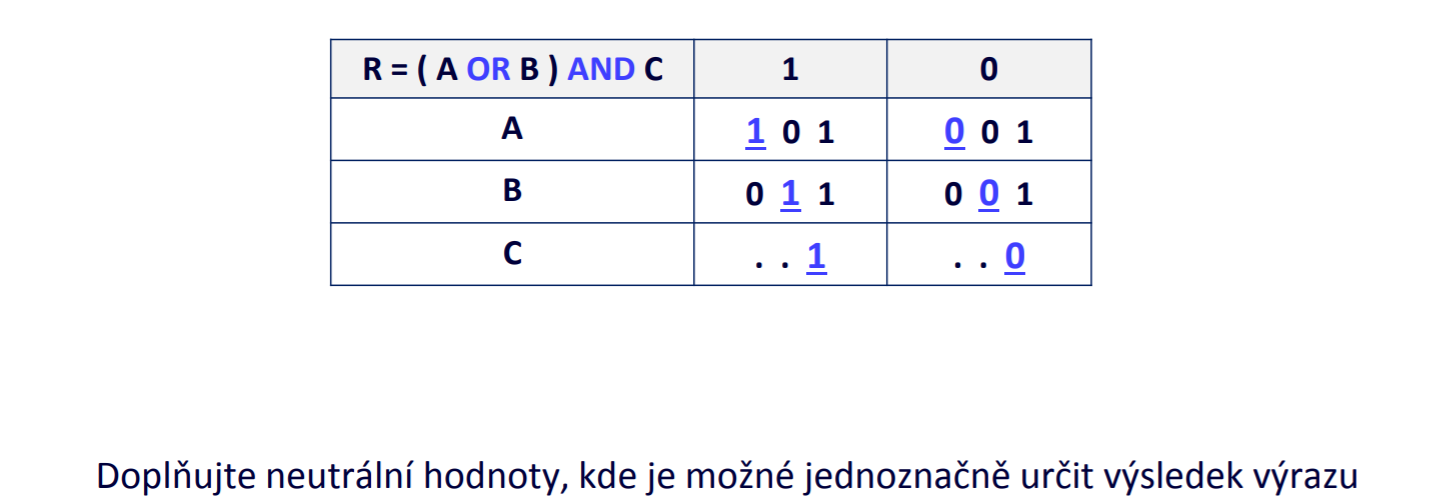
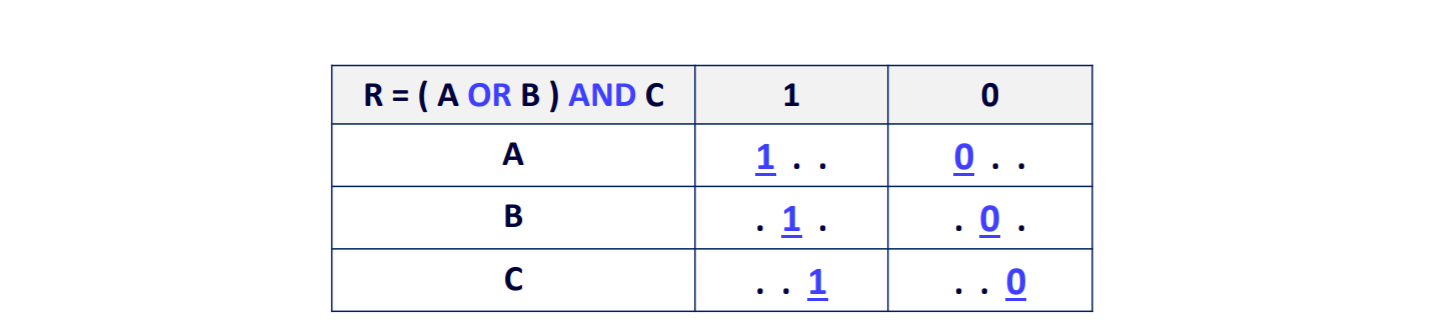
### MCC





### MC/DC





Vytvoříme diagonálu nul a jedniček. Každá proměnná se bude vyskytovat v TRUE a FALSE hodnotě.

Pokud máme na výběr různé hodnoty volíme ty, co se opakují nebo ty, co jsou nejrelevantnější vůči testům.

Pro první sloupec A OR B musí být 1, jelikož neutrální hodnota pro AND je 1: Možnosti:1 1, 0 1, 1 0. Pro druhý sloupec je přípustná každá dvojice nuly a jedničky: 0 0, 0 1, 1 0, 1 1, jelikož žádná kombinace výraz nezmění.

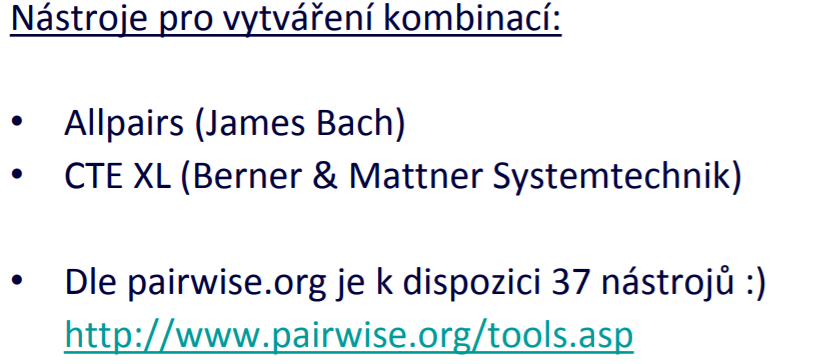
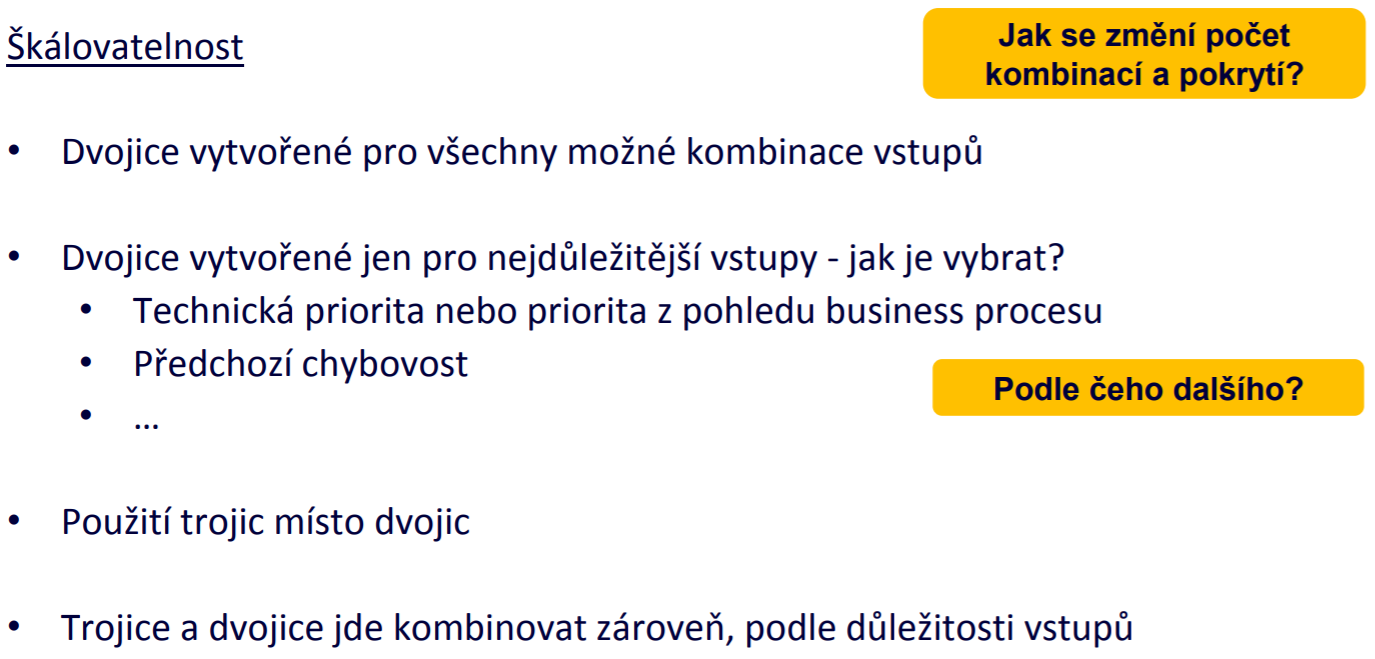
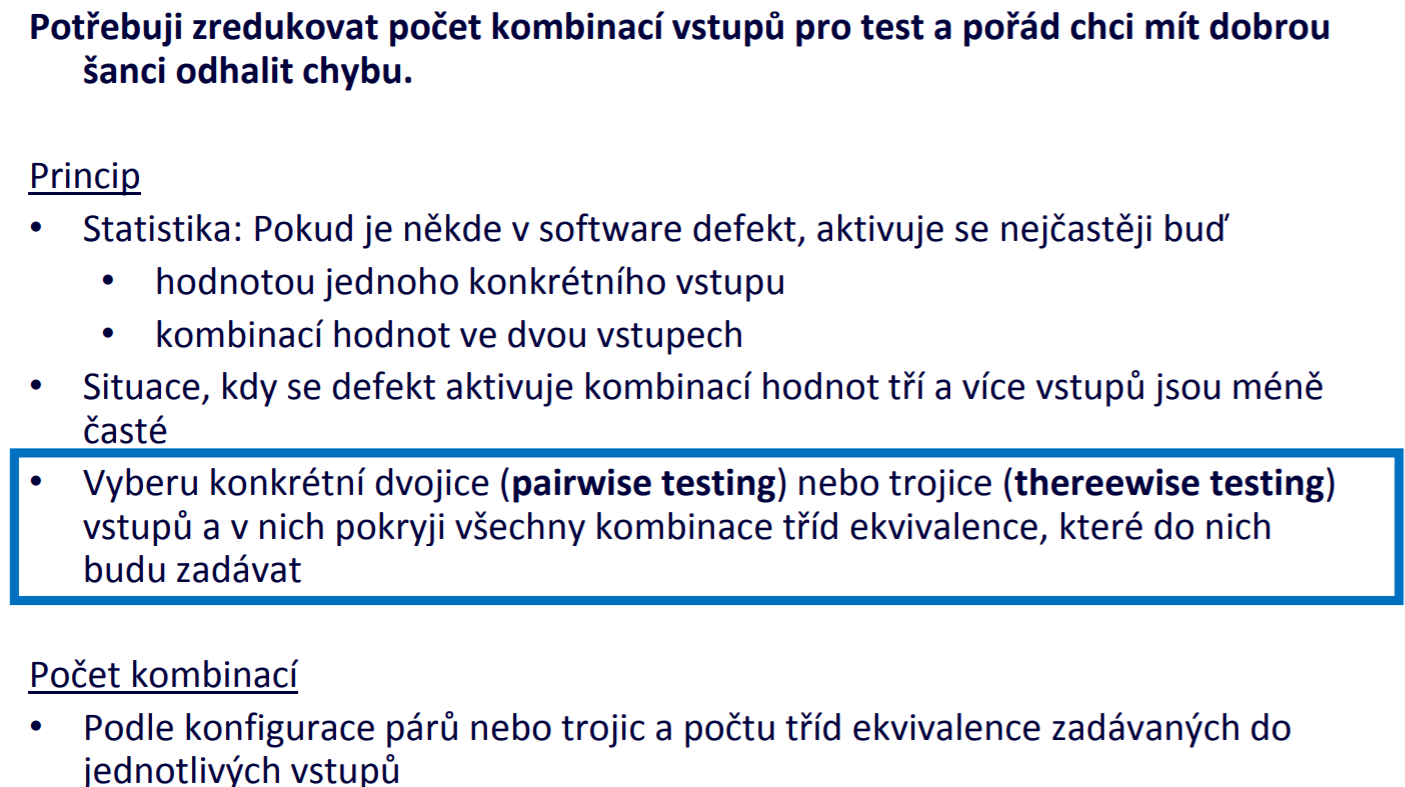
Doplníme **neutrální hodnoty**, které určitě neovlivnily celkový výraz. Dosáhneme tím, že každá modrá proměnná určitě ovlivnila výraz.



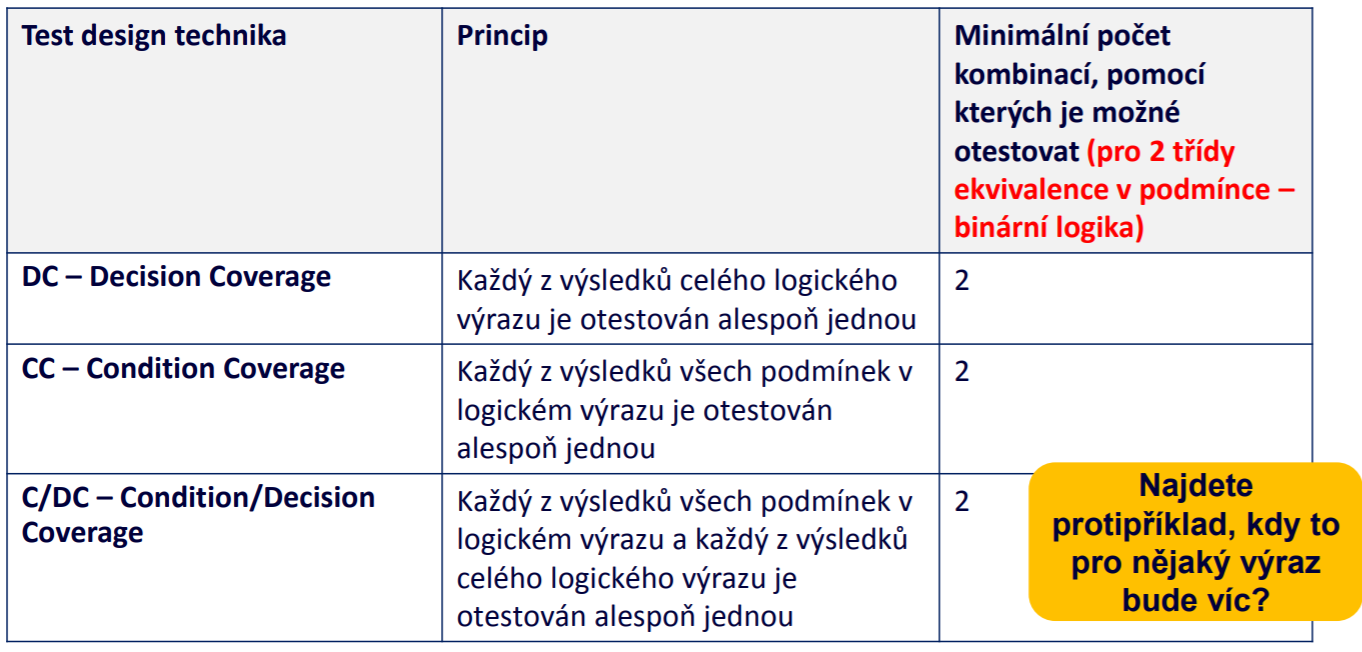
Šedé testy se nám opakovaly, můžeme je zredukovat. Výsledkem jsou 4 testy místo 23=8

### Pairwise

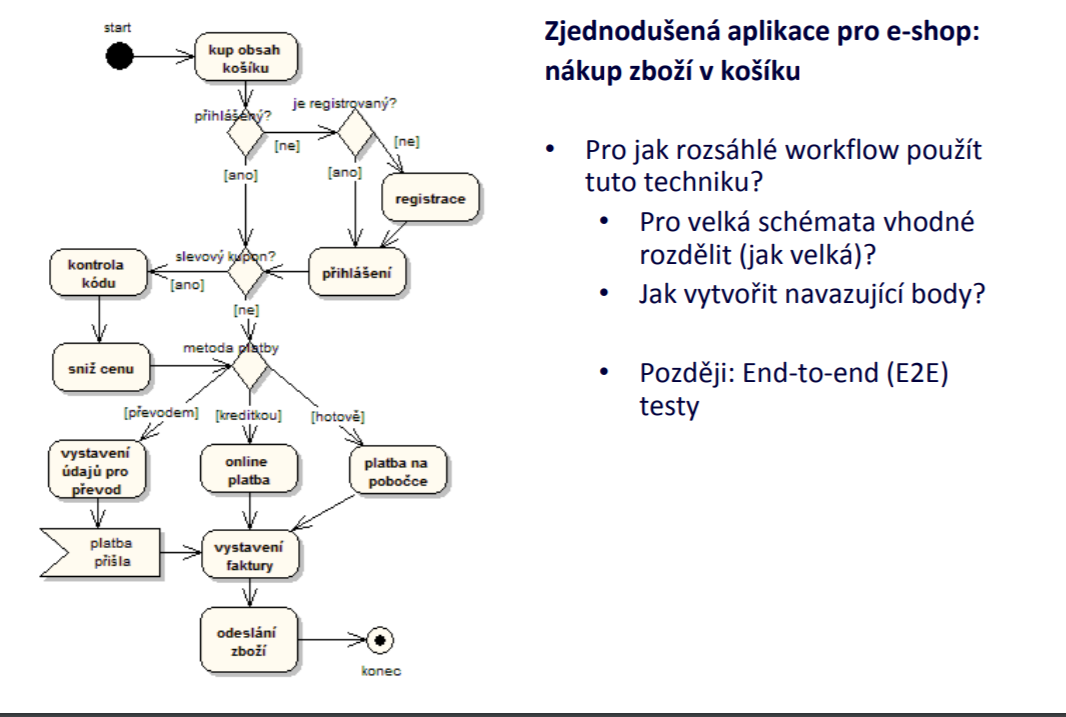
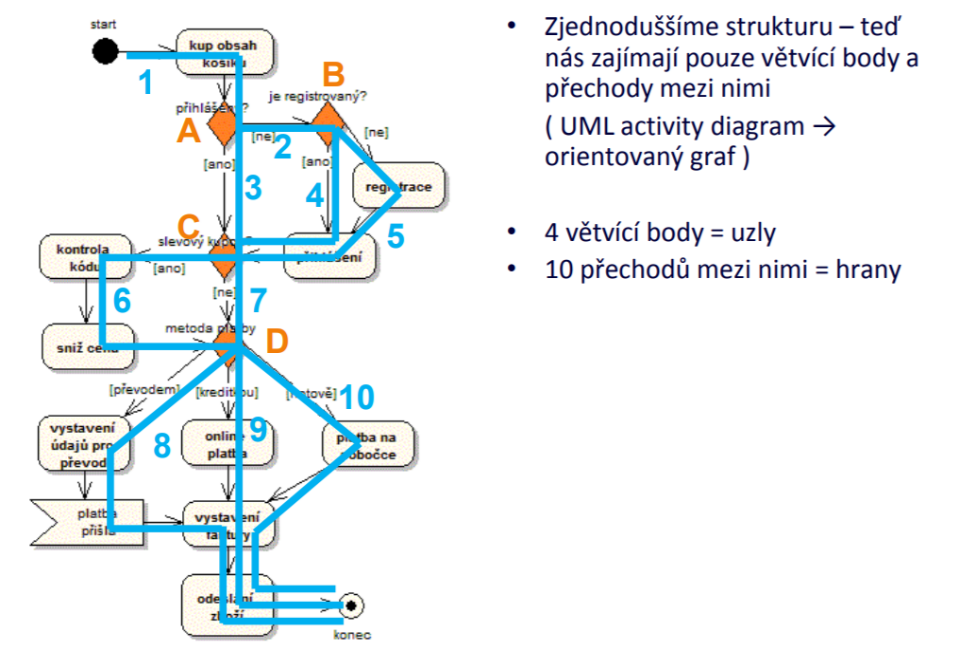
* Vytvoření všech dvojic vstupních parametrů.

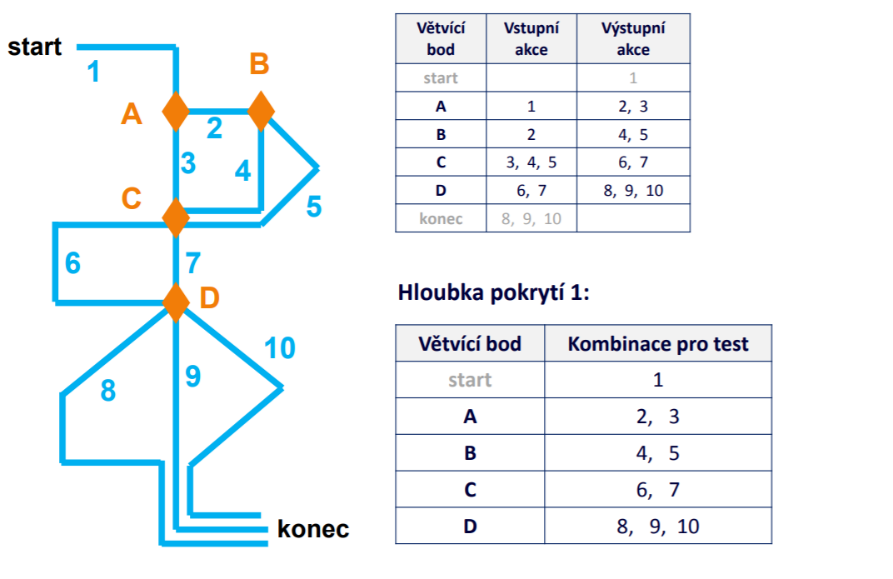


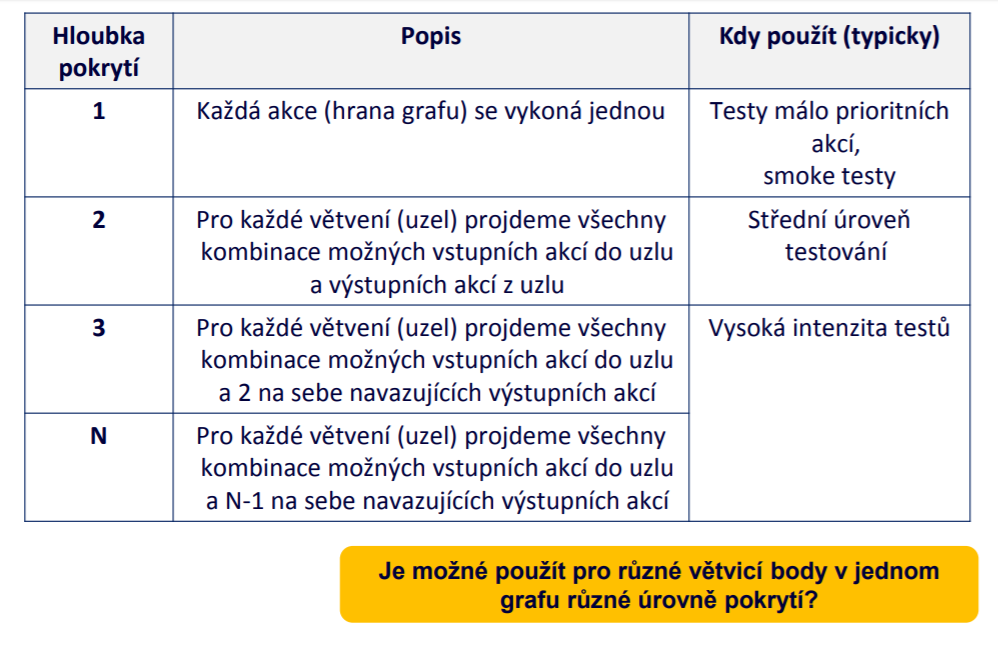
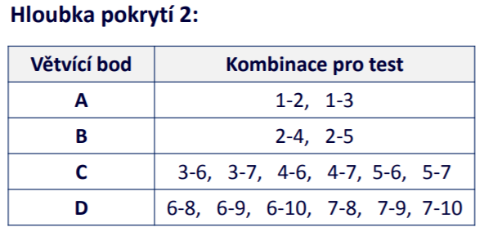
### DC, CC, C/DC

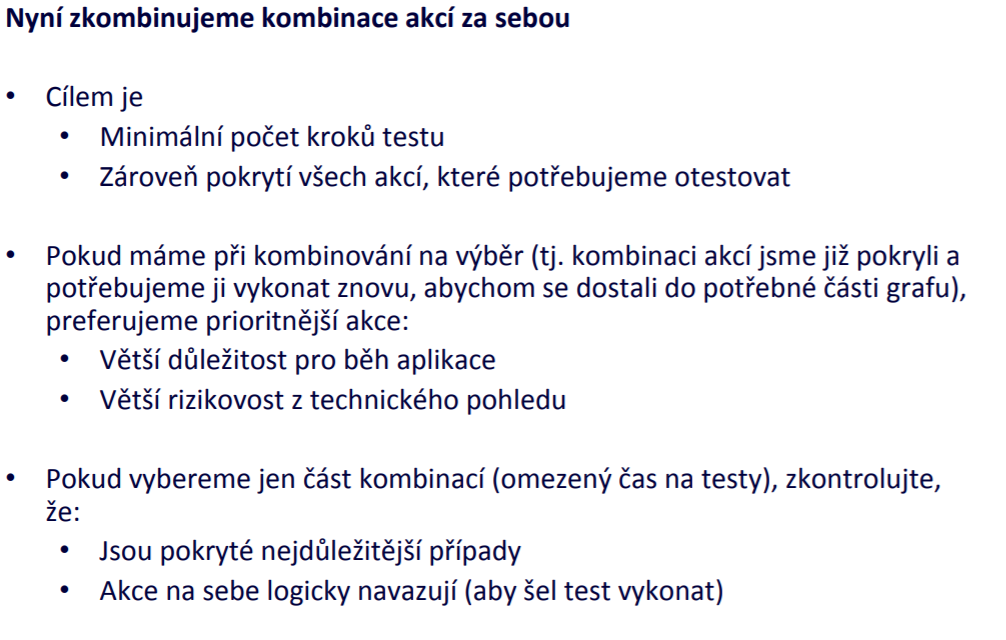
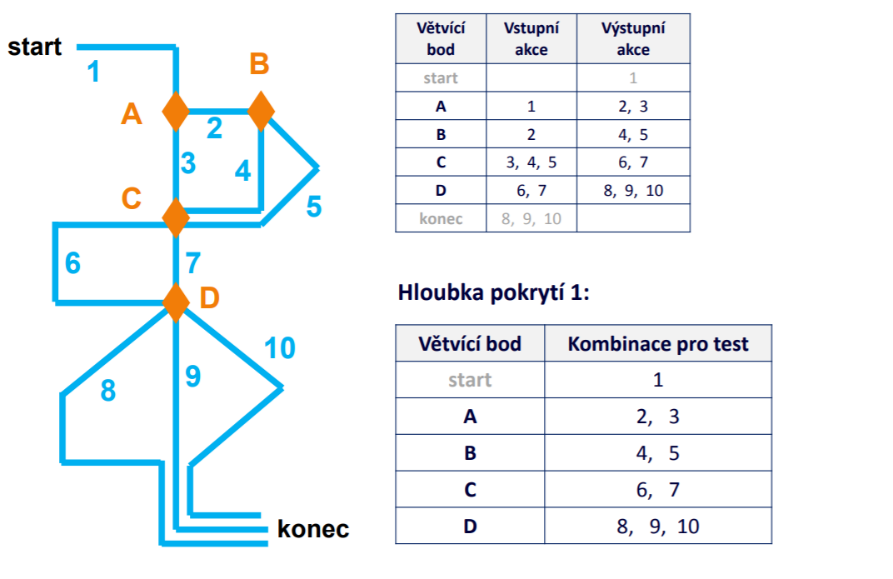
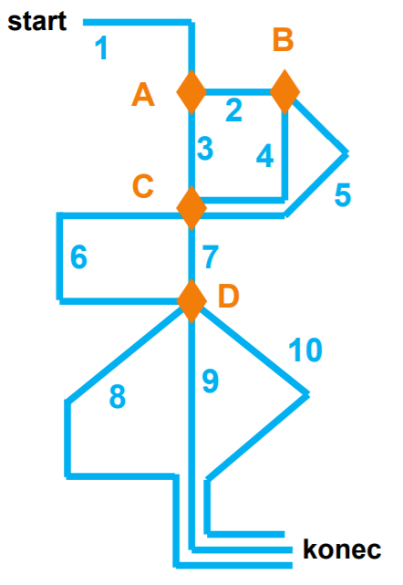


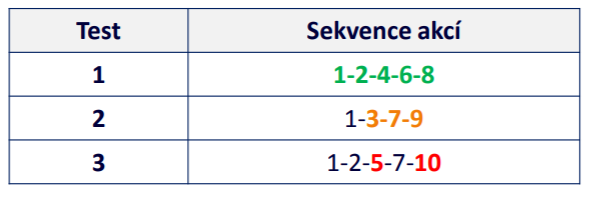
## Průchody a pokrytí cest

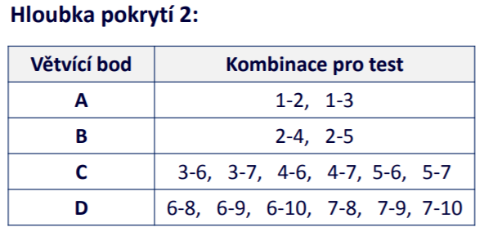


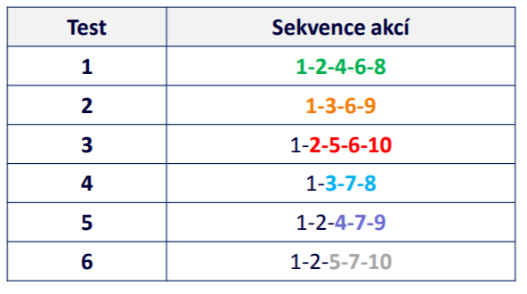












# Testovací strategie

**Plán testování** považuji v návrhu postupu testování softwaru za stěžejní dokument pro celý proces. Při realizaci návrhu jej doporučuji vytvořit jako první dokument před samotným začátkem testování softwaru. Při realizaci postupu testování softwaru musí být jakousi „biblí“ pro každého člena testovacího týmu a musí obsahovat odpovědi na veškeré otázky ohledně procesu testování daného software. Pokud na nějakou otázku z jakýchkoliv důvodů nemůže být odpověď uvedena přímo v dokumentu, bude obsahovat odkazy na místa (nebo osoby), kde se odpověď nalézá.

Plán testování obsahuje především rozsah postupu testování software, [definuje druhy a kategorie testů](http://testovanisoftwaru.cz/category/druhy-typy-a-kategorie-testu/), stanovuje harmonogram prací.

Plán testování musí minimálně obsahoval:

* **Cíle testování** – definice toho, co očekáváme přesně od provedení testů nad softwarem. Někdy to může být jen ověření nové funkčnosti, jindy naopak kompletní otestování celé aplikace.
* **Seznam plánovaných oblastí k testování** – Přestože by cílem testování při realizaci měly být všechny funkce software, je nutné vytvořit seznam těchto oblastí. K těmto oblastem také navrhuji přiřadit prioritu, která bude určovat, v jakém pořadí budou oblasti otestovány.
* **Kategorie testů** – [kategorie testů](http://testovanisoftwaru.cz/category/druhy-typy-a-kategorie-testu/), které budou využity během testovacího cyklu. Případně jejich využití v různých úrovních testování
* **Seznam testovacích případů** – seznam unikátních identifikátorů testovacích případů. Navrhuji je seřadit podle pořadí, v jakém budou spouštěny
* **Definice rizik pro testování** – definice hlavních rizik, které mohou znemožnit testování. Jejich podrobný seznam s ohodnocením a návrhem řešení těchto situací je zpracován v analýze rizik
* **Požadavky na testovací data** – definice dat, které jsou nezbytná pro provádění testů.

Mimo výše uvedené oblasti navrhuji umístit do plánu testování také **harmonogram plánovaných testů**. Harmonogram umožňuje čtenářům plánu testování udělat si velmi dobrou představu o tom, jaké činnosti na sebe kdy navazují. Nevýhodou je však situace, kdy se harmonogram mění i během realizace postupu testování softwaru. Udržovat pak dokument stále aktuální může být obtížné. Navrhuji proto poznamenat v plánu testování, že harmonogram je pouze orientační a nelze jej považovat za definitivní. Poznámka proto musí obsahovat sdělení, že je nutné, ověřit si aktuálnost údaje v harmonogramu u vedoucího testovacího oddělení (autora dokumentu).

Dále navrhuji dbát na přehlednost a srozumitelnost dokumentu plánu testování, jelikož je ze své podstaty určen pro všechny pracovníky, kteří se podílejí na vývoji softwaru. Rozhodně nedoporučuji uvádět odborné termíny z oblasti testování softwaru bez detailnějšího popisu.

# Testovací data

Další oblastí v testování podnikových aplikací, jíž je třeba se zabývat, jsou testovací data. Potřeba testovacích dat je definovaná na základě test analýzy z jednotlivých testovacích scénářů a týká se všech úrovní testování (viz V-model). Správná testovací data jsou klíčová pro úspěšné provedení testů. S ohledem na architekturu aplikací platí, že je třeba testovací data připravovat konzistentně a vzájemně mezi jednotlivými systémy a s ohledem na cíle testování.

Jak mohou testovací data v testovaných systémech vzniknout? Cest je několik. Jednou možností je „kopie“ z produkčního prostředí. Zde je ovšem třeba zajistit vhodnou anonymizaci dat a také uvážit to, že testovací data nemusejí mít potřebnou variabilitu. Další možností je pořízení syntetických dat přes testovanou aplikaci, nebo přes primární systémy. V praxi se obě varianty vhodně kombinují.

Technicky lze data do daného systému pořídit např. pomocí DB insertů, zde je ale riziko nedodržení všech nutných referencí mezi DB tabulkami. Vhodnější je tedy pořídit data uživatelsky, zde ovšem je třeba, aby aplikace byla v této části funkční, což při vývoji nemusí být zajištěno a pak je tato metoda poměrně pracná. Zde lze doporučit využití nástrojů pro automatizované testování.

Automatizované testování softwaru

je stále v praxi velmi podceňovanou disciplínou. Přitom jeho využití sebou přináší výrazné úspory prostředků a samozřejmě zvýšení kvality výsledného produktu. Automatizovat lze však nejen samotnou exekuci vybraných manuálních testů, ale také části nebo dokonce celý proces testování softwaru. Některé nástroje, které porovnávají výsledky provedených testů, instalují a konfigurují aplikaci nebo vytváří z analýzy testovací případy. Pokud jsou pro daný software vytvořené stabilní manuální testovací případy, které se opakovaně vykonávají, je nasnadě otázka, zda by nebylo možné tyto vykonávat automaticky (tedy bez zásahu lidského faktoru). Oblast automatizace testů je čím dál populárnější a často vyhledávanou možností, jak software testovat. Proto jsem se rozhodl se zde zmínit i o možnostech automatizace testů a nastínit tak současné postupy automatizovaného testování software.

V této části bych se rád zmínil o možnosti automatizace provádění testů softwaru. Uvedu jejich hlavní význam, realizaci a porovnání výhod a nevýhod jejich použití.

Význam automatizace testování softwaru

Hlavním cílem **automatizace testů softwaru** je časová úspora při jejich spouštění. Obecně lze říci, že automatizace testů softwaru má za účel usnadnit a zefektivnit provádění postupu testování softwaru. Pokud jsou testy prováděny zcela automaticky, tj. bez možnosti zásahu lidského faktoru, výrazně se tím snižuje možnost chybného provedení postupu testování softwaru a tím také pravděpodobnost bezporuchového provozu softwaru. Automatizované testy lze rozdělit do několika kategorií. Tyto kategorie jsou shodné s těmi uvedenými pro manuální zkoušky. Tato skutečnost vyplývá z faktu, že automatizované testy ve většině případů jednoduše zautomatizují provádění manuálních testů.

Použití automatizovaných testů však má své výhody i nevýhody. Porovnání některých pro a proti uvádím v na konci stránky