

# IV113 Validace a verifikace

## Testování

doc. RNDr. Jiří Barnat, Ph.D.

**Technické vyšetřování testovaného produktu prováděné za účelem poskytnutí kvalitativních informací zainteresovaným subjektům.**

## **Technické**

- experimentování
- logika a matematika
- modelování
- SW nástroje pro samotné testování
- pomocné SW nástroje

## **vyšetřování**

- organizované a důkladné hledání
- sebekritické a vyzývající

## **testovaného produktu**

- samotný kód
- neoddělitelná data
- dokumentace a specifikace
- HW
- a dalších věcí, které jsou součástí dodávky zákazníkovi

## **prováděné za účelem poskytnutí kvalitativních informací**

- viz dále

## **zainteresovaným subjektům.**

- někdo, kdo má zájem na tom, aby testování bylo smysluplné (šéf testovacího týmu)
- někdo, kdo má zájem na tom, aby produkt byl úspěšný (manažer produktu)
- čí zájem je možné/žádoucí ignorovat

## Mise

- Proč testujeme? Co se snažíme testováním dosáhnout?

## Strategie

- Jakým stylem máme postupovat, abychom dosáhli cíle?

## Problém orákula

- Jak vlastně poznáme, že test proběhl úspěšně?

## Neúplnost

- Uvědomujeme si, že testováním nelze potvrdit absenci chyby?

## Míra

Můžeme testování ukončit?

- Jaký je testovací plán? Kolik už je otestováno?

## Nejčastější mise

- Detekce chyb.
- Identifikace faktorů snižující kvalitu produktu.

## Jiné cíle testování

- Vytvoření podkladů pro rozhodnutí, zda je produkt již dost dobrý na to, aby byla zahájena jeho distribuce.
- Nakolik se produkt liší (například v ovládní) od produktů momentálně dostupných na trhu?
- Posouzení, zda produkt pokrývá požadavky zadavatele.
- Je provázání souvisejících funkcí software logické a dostatečné?
- ...

## Jiné cíle testování – pokračování

- Podpořit/nabourat manažerská rozhodnutí čísky.
- Odhadnout cenu nabízené podpory produktu po jeho uvolnění.
- Ověřit kompatibilitu a interoperabilitu vůči jiným produktům.
- Nalézt bezpečné scénáře použití produktu.
- Potvrdit soulad se specifikací.
- Certifikovat daný standard.
- Minimalizovat rizika vedoucí k právním dopadům.
- Vyhodnotit produkt pro jiného zadavatele.
- ...

## Strategie testování

Strategie je plán, jak naplnit misi testování v kontextu konkrétního projektu.

**Příklad:** Uvažme program, který provádí výpočty ala tabulkový procesor v následujících 4 kontextech

- a) počítačová hra
- b) rané stádium vývoje komerčního produktu (mise: identifikace problémových míst, první zpětná vazba programátorům)
- c) pozdní stádium vývoje komerčního produktu (mise: pomoci projektovému manažeru rozhodnout, zda je produkt hotov)
- d) ovladač ozařovacího zařízení na léčbu rakoviny

**Otázka:**

- Budeme postupovat v různých případech stejně?

## Faktory ovlivňující výběr strategie

- Jaká je mise v jednotlivých kontextech?
- Jak agresivně budete hledat chyby?
- Jaké chyby jsou méně důležité než jiné a proč?
- Jak důkladně budete dokumentovat proces testování?

## Diskuze

- Předpokládejme, že dle specifikace má program vstupní pole, na kterém očekává číselné hodnoty (program provádí výpočty). Má smysl testovat chování programu, pro situace, kdy na vstupu nejsou čísla, ale písmena (situace mimo specifikaci).

## Problém orákula

Orákulum (v kontextu testování) je princip nebo mechanismus, kterým jsme schopni rozeznat, že něco není tak, jak by mělo být (problém).

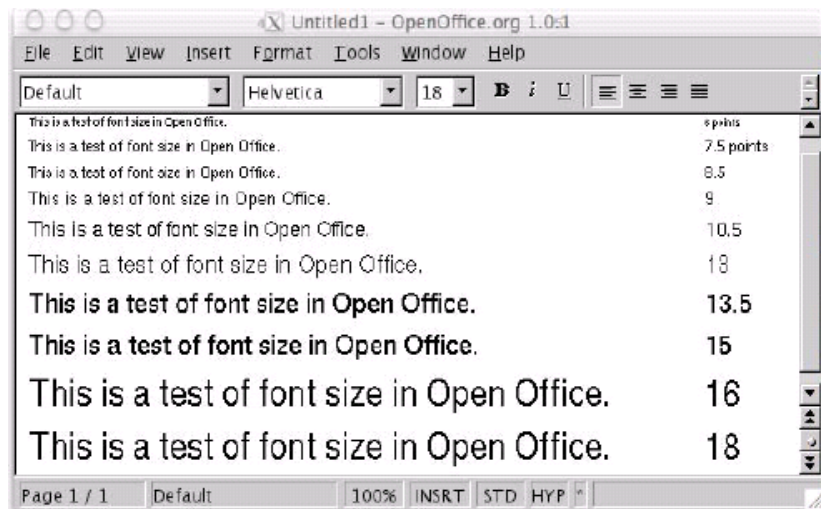
## Fakta

- Tvrdí-li tester, že test neprokázal nedostatky, neznamená to, že je produkt v daném směru bezchybný.
- Výsledek každého testu může být “test proběhl v pořádku”, záleží pouze na volbě orákula.

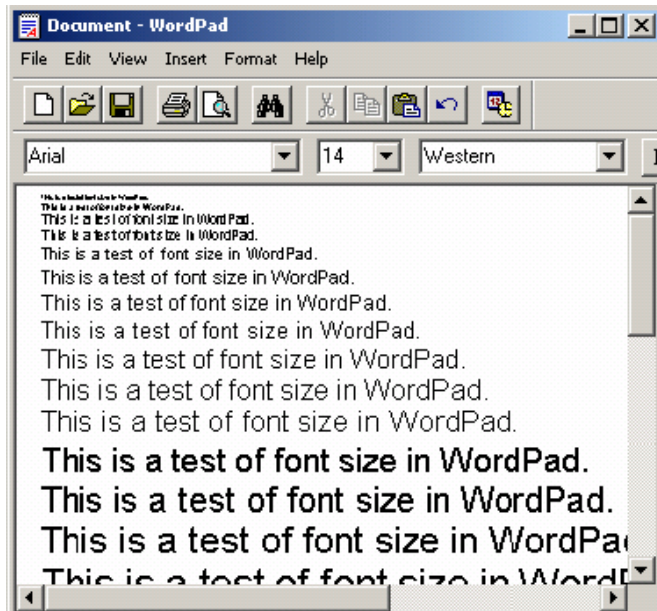
## Příklad

- Fungují správně velikosti písem v programech OpenOffice, Word PAD, Word?

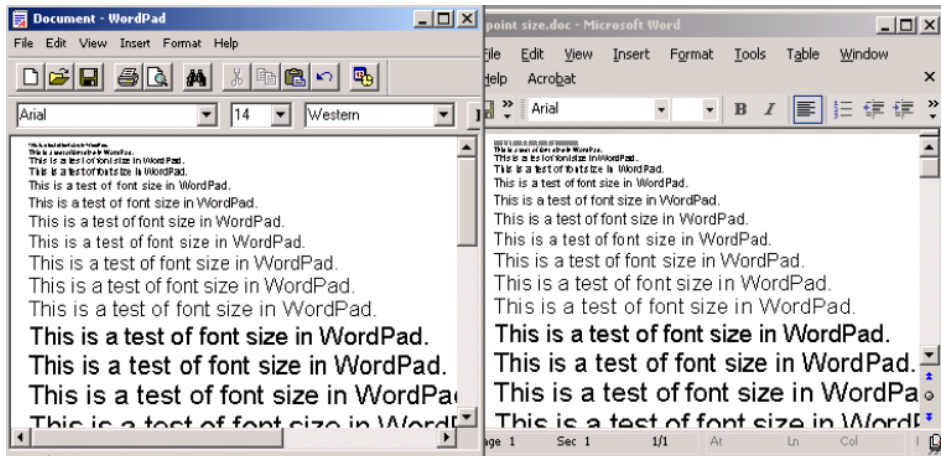
# Příklad – OpenOffice 1.0



# Příklad – Word PAD



# Příklad – Word PAD versus MS Word





## Otázky

- Je pozorovaný rozdíl velikostí bug ve Word Padu?
- Je pozorovaný rozdíl velikostí bug v MS Wordu?
- Je pozorovaný rozdíl velikostí vůbec bug?

## Možné závěry

- Nevíme, jestli jsou velikosti písma správně, ale při porovnávání Word Padu a MS Wordu, raději věříme MS Wordu.
- Pro Word PAD není třeba lpět na přesných standardech typografie.
- Pro Word PAD je pozorovaný rozdíl (možná) bug, ale není to problém.

## Možný (pragmatický) pohled na věc

- Je/Není to bug?  $\implies$  Je/Není to problém?
- Je třeba znát kontext, do kterého bude testovaný produkt zasazen, případně metriky podle kterých produkt posuzuje zákazník.
- Zjednodušení procesu testování za cenu jistého rizika.

## Zjednodušení

- Vynechání testů, které zřejmě neodhalí žádné problémy.
- Vynechání testů, které zřejmě odhalí pouze nezajímavé problémy.

## **Kolik víme o typografii?**

- Definice bodu (point) je nejasná.  
(<http://www.oberonplace.com/dtp/fonts/point.htm>)
- Absolutní velikost písma není lehké změřit.  
(<http://www.oberonplace.com/dtp/fonts/fontsize.htm>)

## **Nejasnost a náročnost posouzení výsledku testu**

- Jak přesně musí velikost být v souladu se standardem, abychom prohlásili, že je velikost písma korektní?
- Kompletní shromáždění faktů a jejich vyhodnocení je příliš náročné/zdlouhavé.
- Při rozhodování o výsledku testu se používají heuristiky.

## Heuristika

- Je postup, který umožní zjednodušit a snáze vyřešit problém rozhodnutí.
- Neobsahuje žádnou skrytou znalost.
- Rada/návod/doporučení na základě kontextu.
- Negarantuje správnost rozhodnutí.
- Různé heuristiky mohou vyústit ve vzájemně rozporná rozhodnutí.

## Nevýhody

- Při nesprávném použití mohou být na škodu věci.
- V obecné rovině jsou heuristiky subjektivní.

## Konzistence

- Dobrá heuristika pro rozhodování o výsledku testu.

## Konzistence s

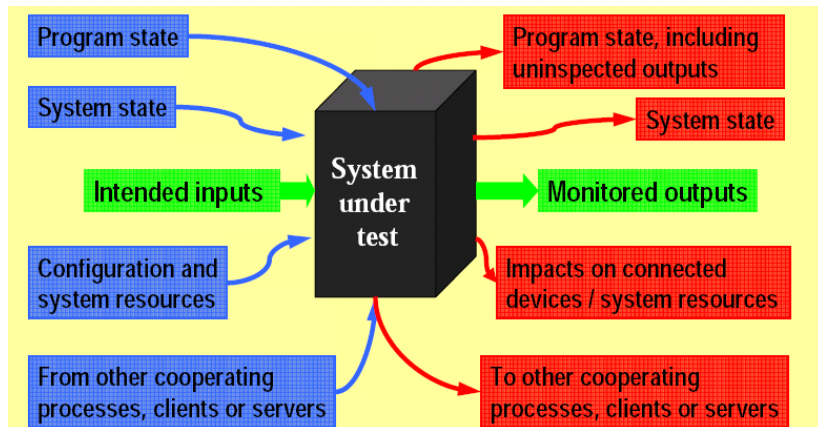
- ostatními funkcemi produktu, porovnatelnými produkty, historií, image producenta, různými prohlášeními a reklamou, specifikací či standardy, očekáváním uživatelů, účelem produktu, ...

## Výhody

- Je dostatečně objektivní.
- Je snadno popsatelná (bug report).

# Širší kontext testu SW produktu

Důvody pro selhání produktu jsou často nad rámec vstup-výstupního chování produktu. Je nutné produkt testovat i nad tento rámec, což v kontextu problému orákula znamená rozhodovat i dle nepřímých výstupů.



## Slepota z nepozornosti

- Lidský tester neuváží do rozhodnutí to, na co nedává pozor (to, co nesleduje).
- Mechanický tester neuváží do rozhodnutí to, co mu nebylo řečeno, aby do rozhodovacího procesu zahrnul.

## Princip neurčitosti

- Zapojením mnoha diagnostických prostředků se zkresluje chování testovaného produktu.

## Důsledek

- V rámci testu nelze z praktického hlediska sledovat všechny možné aspekty.

## Motivace

- Automatizace procesu vylučuje lidské chyby.
- Automatizací získáme opakovatelnou proceduru.
- Větší rychlost provádění jednotlivých testů.

## Problém automatizace

- Je třeba (mimo jiné) automatizovat rozhodovací proces.
- Umíme to? (Částečně)

## Standardní způsob automatizace rozhodovacího procesu

- Definujeme zdroj/soubor očekávaných výstupů, pokud možno včetně nepřímých výstupů.
- Příklad: MS Word je zdrojem výstupů pro MS Word PAD
- Test je úspěšný pokud výstup testovaného produktu odpovídá (jedna k jedné) výstupu dle souboru očekávaných výstupů.

## Problém definice shody

- Jak uloším výstup MS Wordu, tak abych ho mohl porovnávat s výstupem z testovaného Word Padu?
- Je přijatelná 99% shoda? Jak definovat % shody?

## Falešná hlášení o chybách

- Neshoda neznamená nutně chybu, může být způsobena například přechodem na novější verzi.
- Změny v návrhu systému vynucují změny ve zdrojovém souboru, či v testech samotných.

## Nenalezené chyby

- Shodná chyba v testovaném produktu a v souboru očekávaných výstupů.
- Neúplnost zdrojových dat, viz slepota z nepozornosti.

## Metody míry testování

## Pokrytí

- Množina testováním prověřených entit programu (entity: řádky kódu, podmínky, vstupní data, větve programu, ...)
- Identifikaci netestovaných částí kódu.

## Pokrytí jako míra

- Možný testovací plán je dosáhnout daného procenta pokrytí výsledného produktu.
- Procento pokrytí stávajícími testy je pak možné chápat jako míru jak daleko jsme v testovacím plánu.
- Pro manažery a vedení projektu je dobré umět změřit kolik z celkového objemu testování bylo provedeno, případně kolik zbývá.

## Principiální nedostatky pokrytí

- Nepostihne zajímavá vstupní data.
- Nepostihne kód, který není součástí produktu (knihovny, ovladače, atd.)
- Netestuje produkt v kontextu běžícího OS systému (například možné okamžiky, ve kterých dochází k HW/SW přerušení a vykonání odpovídající obslužné rutiny.)

## Používání pokrytí jako míry

- Úplné pokrytí negarantuje kvalitu produktu.
- Stimuluje tendenci preferovat kvantitu před kvalitou.
- Zavádějící, navozuje falešný pocit bezpečí.

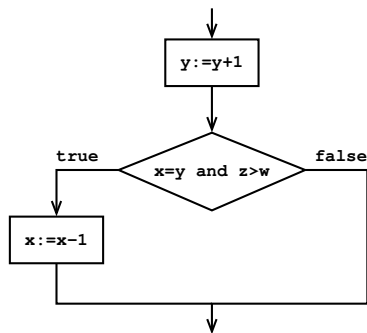
## Příklad

```
Input A      // program accepts any
Input B      // integer into A and B
Print A/B
```

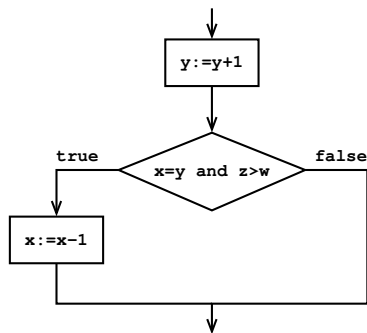
## Pozorování

- Snadno dosáhneme úplného pokrytí.
- Například:  
    input: 2,1  
    output: 2
- Tento test neodhalí skrytý “bug”!

# Kritéria pokrytí pro Control-Flow grafy

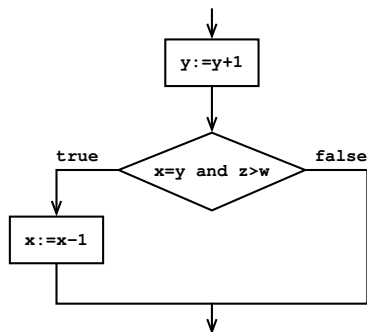


Existují různá kritéria pokrytí Control-Flow grafu.



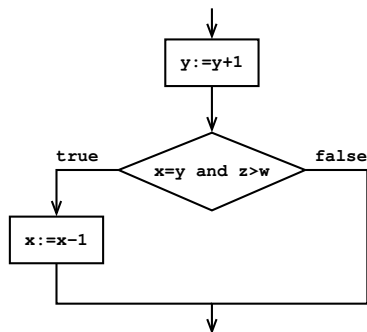
## Statement coverage – pokrytí výrazů

- Každý výraz (přiřazení, vstup, výstup, podmínka) je proveden alespoň v jednom testu.
- Sada testů pro dosažení pokrytí:  $(x = 2, y = 1, z = 4, w = 3)$



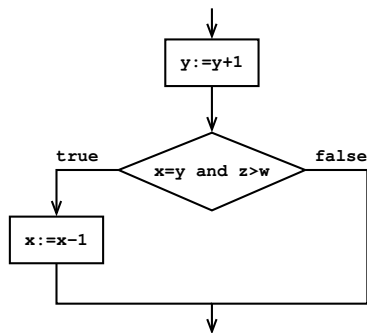
## Edge coverage – pokrytí hran

- Každá hrana CF grafu je provedena alespoň v jednom testu.
- Sada testů pro dosažení pokrytí:  $(x = 2, y = 1, z = 4, w = 3)$ ,  $(x = 3, y = 3, z = 5, w = 7)$



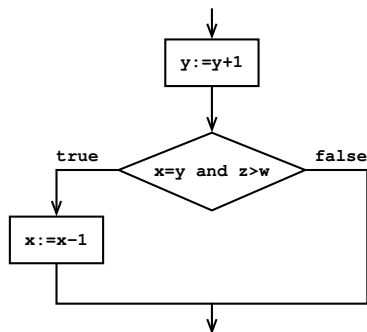
## Condition coverage – pokrytí podmínek

- Každá podmínka je Boolovskou kombinací **elementárních podmínek**, například  $x < y$  nebo  $\text{even}(x)$ .
- Pokud je to možné, každá elementární podmínka je alespoň v jednom testu vyhodnocena na TRUE a alespoň v jednom testu vyhodnocena na FALSE.



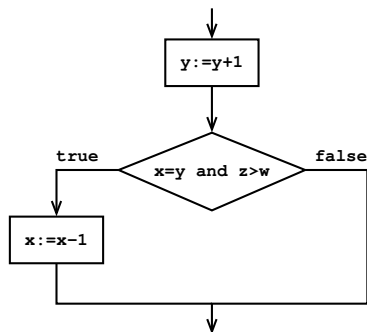
## Condition coverage – pokrytí podmínek

- Sada testů pro dosažení pokrytí:  $(x = 3, y = 2, z = 5, w = 7)$ ,  $(x = 3, y = 3, z = 7, w = 5)$
- V obou případech je podmínka ve výrazu IF vyhodnocena na FALSE.



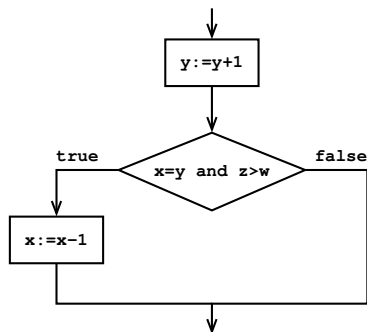
## Edge/Condition coverage – pokrytí hran a podmínek

- Pokrytí proveditelných hran a podmínek zároveň.
- Sada testů pro dosažení pokrytí:  $(x = 2, y = 1, z = 4, w = 3)$ ,  $(x = 3, y = 2, z = 5, w = 7)$ ,  $(x = 3, y = 3, z = 7, w = 5)$
- Je uvedená sada testů nejmenší možná?



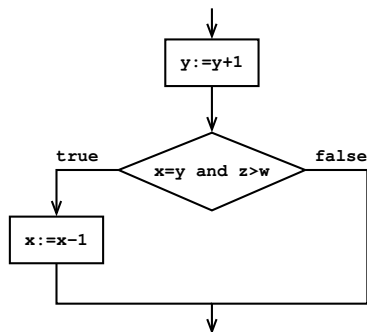
## Multiple condition coverage – násobné pokrytí podmínek

- Každá Boolovská kombinace hodnot TRUE/FALSE, která se může objevit v nějaké rozhodovací podmínce, se musí objevit v provedení alespoň jednoho testu.



## Multiple condition coverage – násobné pokrytí podmínek

- Sada testů pro dosažení pokrytí:  $(x = 2, y = 1, z = 4, w = 3)$ ,  
 $(x = 3, y = 2, z = 5, w = 7)$ ,  $(x = 3, y = 3, z = 7, w = 5)$ ,  
 $(x = 3, y = 3, z = 5, w = 6)$
- Exponenciální růst počtu testů.



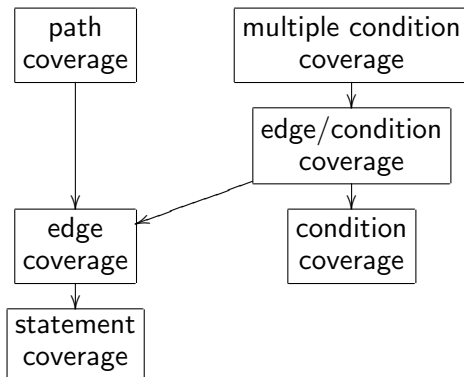
## Path coverage – Pokrytí cest

- Každá proveditelná cesta je provedena alespoň v jednom testu.
- Počet cest je obrovský, přítomnost cyklu, může vyústit v nekonečný počet cest.

- Kritérium A **zahrnuje** kritérium B, značeno  $A \rightarrow B$ , pokud dosažením pokrytí typu A také garantuje pokrytí typu B.

# Hierarchie kritérií pokrytí pro Control Flow graf

- Kritérium A **zahrnuje** kritérium B, značeno  $A \rightarrow B$ , pokud dosažením pokrytí typu A také garantuje pokrytí typu B.



## Pokrytí a průchody cyklem

- Všechna zmíněná kritéria (s výjimkou pokrytí cest) neřeší počet průchodů tělem cyklu.
- V případě existence zanořených cyklů je systematické testování různých způsobů průchodů cykly komplikované.

## Ad hoc strategie pro testování cyklů

- Prověř případ, kdy se tělo cyklu přeskočí.
- Prověř případ, kdy se tělo cyklu provede přesně jednou.
- Prověř případ, kdy se tělo cyklu provede očekávaným počtem opakování.
- Pokud je známa hranice  $n$  na počet provedení těla cyklu, proveď případ, kdy je tělo cyklu provedeno  $n - 1$ ,  $n$ , a  $n + 1$  krát.

## Motivace

- Použití nedefinovaných proměnných.
- Mohou být cesty v programu, na kterých je nějaká proměnná nastavena za určitým úmyslem, ale posléze je hodnota této proměnné zneužita k jinému účelu.
- Control Flow kritéria nezaručují zahrnutí testů pokrývajících popsaný případ.

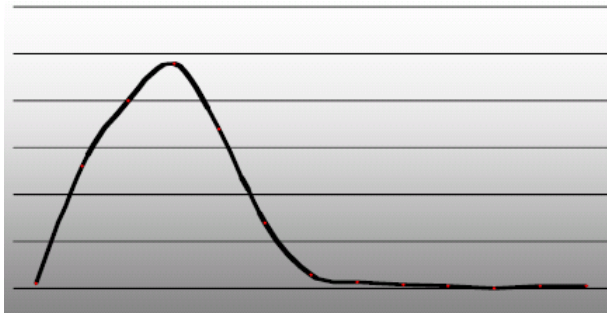
## Data Flow pokrytí

- Pokrytí všech míst programu, ve kterých je daná proměnná použita, ne však nutně definována podél všech cest v Control-Flow grafu.

## Týdenní statistiky

- Počet nově odhalených chyb
- Počet opravených chyb
- Podíl nalezených vůči opraveným chyb

## Ukázka křivky



## Pozorování

- Množství nalezených chyb vykazuje Weibullovo pravděpodobnostní rozložení
- Metoda míry provedeného testování, resp. množství zbývajících objemu testování.
- Metoda určování data uvolnění produktu na trh.

## Postup

- V okamžiku, kdy dojde rozložení za vrchol, lze za předpokladu znalostí parametrů Weibullova rozložení odhadnout, kdy pravděpodobnost odhalení další chyby v produktu klesne pod danou mez.
- Parametry rozložení ovlivňují “šířku” a “výšku/strmost” kopce.
- $F(x) = 1 - e^{-ax^{-b}}$  pro  $x > 0$

## Fakta způsobující nepřesnost výše uvedené metody

- Testování nesleduje očekávaný způsob používání produktu.
- Pravděpodobnost nalezení různých chyb není stejná.
- Oprava chyb může způsobit nové chyby.
- Chyby nejsou nezávislé.
- Počet chyb v produktu se mění (není dána počáteční fixní hodnota).
- Samotné zanášení chyb do systému sleduje Weibullovo rozložení.
- Epochy v testování (různé testovací postupy) jsou nezávislé.
- Parametry rozložení nejsou dány.

## Závěr

- Závěry vycházející z uvedeného rozložení jsou platné pouze pro velké projekty a i tak jsou často zavádějící.

## První fáze

- Snaha o strmější stoupání a rané vyvrcholení křivky.
- Jakmile se dosáhne vrcholu křivky, lze odhadnout tvar křivky a udělat první odhady data uvolnění produktu.

## Dopady

- Spouštění testů nad částmi produktu, o kterých se ví, že jsou vadné, nebo nedokončené.
- **Preferuje se hledání a reportování snadných chyb, namísto hledání těch skutečně závažných.**
- Důraz kladen na hledání chyb, ne na vývoj testovacích nástrojů (intenzifikace X extenzifikace)
- Vykazování jedné chyby jako několik chyb menších.
- Opakované vykazování chyb (například různými testery)
- ...

## Druhá fáze

- Počet nalezených chyb za čas by měl klesat.
- Z tvaru křivky lze odvodit kolik chyb za čas by se mělo vykázat.
- Snaha vykázat stabilitu křivky (blízkou nule).

## Dopady

- Opakování úspěšných testů.
- Důraz přesunut od hledání nových chyb k důkladnému popsání nalezených.
- Slučování různých chyb do jedné.
- Odkládání nalezení chyb (např. až na “po milestone”).
- Zatajování/odmítnutí/ztracení chyb!
- Neformální reportování chyb, mimo systém.
- Firemní akce pro testery.
- Programátoři neopravují chyby, dokud je testeři nenahlásí.
- ...

## Neúplnost testování

## Pozorování

- Prostor, který má být prohledán, je obrovský.
- Prostředky a zdroje jsou omezené.

## Co není úplné testování

- Úplné pokrytí
  - každý řádek kódu
  - každé větvení
  - každou sekvenci kódu
- Testeři nenacházejí nové chyby
- Testovací plán je dokončen

## Co je úplné testování

- Na konci procesu testování nejsou skryté (neznámé) nedostatky produktu.
- Pokud se objeví nový nedostatek produktu, testování nemohlo být úplné.

## **V časové tísní se většinou pouze**

- Analyzují výsledky testů.
- Řeší problémy.
- Popisují chyby.

## **V časové tísní nezbývá čas na**

- Návrh testů.
- Provádění testů.
- Vývoj testovacího SW.
- Revize, inspekce proběhlých testů.
- Dokumentace testů.
- Automatizace testů.
- ...

## **Pozorování**

- Čas potřebný pro úkony spjaté s testováním je výrazně větší, než čas který je k dispozici.

**Možných testů je velmi mnoho (až nekonečně mnoho).**

**Provést všechny možné testy znamená:**

- Otestovat všechny možné hodnoty na vstupu každé vstupní proměnné.
- Otestovat všechny možné kombinace vstupů všech vstupních proměnných.
- Otestovat každý možný běh systému.
- Otestovat každou možnou konfiguraci HW a SW, včetně konfigurací hypotetických cílových serverů, které jsou mimo vaši kontrolu.
- Otestovat každý způsob, jakým může uživatel produkt použít.

## Šířka datové sběrnice

- Počet testů roste exponenciálně vzhledem k počtu bitů použitých pro reprezentaci dat.
- $n$ -bitů vynucuje  $2^n$  testů.

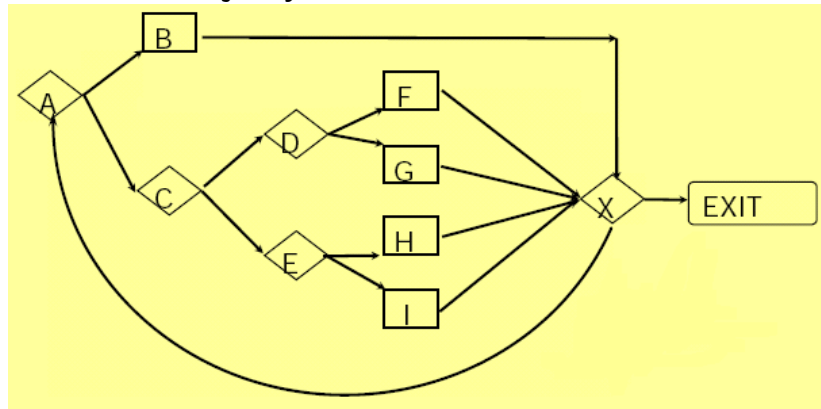
## Další příklady

- Časování akcí
- Neplatné neočekávané vstupy (buffer overflow).
- Editované vstupy
- Velikonoční vejce [\[http://j-walk.com/ss/excel/eastereg.htm\]](http://j-walk.com/ss/excel/eastereg.htm)

## Častá praxe

- “Tohle by žádný uživatel našeho produktu neudělal.”

Uvažme následující systém



**Příklad**

- Kolika způsoby lze dosáhnout `EXIT` ?
- Kolika způsoby lze dosáhnout `EXIT` , jestliže `A` lze navštívit nejvíce 20x?

## Příklad

- V [F] je memory leak, v [B] garbage collector.
- Systém dospěje do neplatného stavu, pouze pokud se cestě s [B] bude dostatečně dlouho vyhýbat.

## Fakta

- Zjednodušené testování “cest” v systému nemusí postihnout kritickou chybu.
- Kritická chyba se projeví za takových okolností, které by se nikdy jednoduchým testem neprověřovaly.
- Problém dlouhých běhů systému.

## Neúplnost

- Testováním nelze prokázat, že systém neobsahuje chyby.
- Existence testovacího plánu brání kreativitě testerů.
- Otestovat všechny možné případy je nemožné.

## Měřitelnost

- Existují metody pro měření postupu ve fázi testování.
- Metody jsou nespolehlivé.
- Posuzování výkonnosti testovací skupiny na základě dané metriky může ovlivnit testování samotné.

Chyby

- Domácí úlohy jsou povinné pro získání závěrečného hodnocení zkoušky stupněm A.
- Termín na vypracování domácích úloh je do konce čtrnáctého dne ode dne zadání.
- Domácí úlohy se odevzdávají vyučujícímu na přednášce, nebo e-mailem na adresu `xbarnat@fi.muni.cz`.

Odborná esej v rozsahu minimálně 1000 slov dávající odpovědi na následující otázky:

- Co je to vlastně chyba?
- Jaký je smysl vedení systému pro správu chyb?
- Jaké informace by měly být součástí záznamu o chybě?
- Jaké mohou být motivace/důvody pro odkládání nápravy chyby nahlášené v systému pro správu chyb?