16. Kategorizace SW chyb, kritéria korektnosti a použitelnosti, spolehlivost SW

1. Softwarová chyba

- Prezentace toho, že program dělá něco nepředpokládaného
- Míra toho, kdy program přestává být užitečný
- Je to nesouhlas mezi programem se specifikací

2. Kategorie softwarových chyb

- **Chyby uživatelského rozhraní**
  - Problémy s funkčností - program nedělá něco, co by měl dělat nebo to dělá nevhodně, lze některé operace provést obtížně
  - Všechny programy mají problémy s funkčností vzhledem k různým uživatelům
  - **Vstupy**
    - Jak lze nalézt, jak program používat (jaká je nápověda)?
    - Jak snadné je ztratit se v programu?
    - Jaké chyby uživatel dělá a kolik ho to stojí času?
    - Co mu chybí?
    - Nutí program uživatele přemýšlet nepřirozeně?
  - **Výstupy**
    - Rychlost je základ interaktivního SW
    - Cokoli co budí dojem pomalého program je problém.
    - Získá uživatel, co potřebuje?
    - Mají výstupy smysl?
    - Může uživatel přizpůsobit výstup potřebám?
    - Lze výstup přesměrovat dle potřeby(monitor, tisk, soubor…)?
- **Chyby omezení**
  - Chyby zpracování výjimek
  - Chyby hraničních podmínek
    - Nejjednodušší hranice jsou numerické
    - Mezní nároky na paměť, za kterých program může pracovat
  - Výpočetní chyby
    - Program ztrácí přesnost během výpočtu vlivem zaokrouhlovacích chyb a ořezání
- **Procesní chyby**
  - Počáteční a jiné speciální stavy (při prvním použití chybí inicializační informace či soubory, Nastaví se skutečně vše do výchozího bodu, vymažou se všechna data, jestliže uživatel provede reset programu?)
  - Paralelní
    - Chyby souběhu
      - Nastávají v multiprocesových sys. a integračních sys.
      - Velmi obtížně se opakují
    - Zátěžové podmínky
      - Chyby velkého objemu - hodně práce za dlouhou dobu

        - Chyby velkého stresu - hodně práce v daném okamžiku
- **Chyby vedení**
    - Hardware - program posílá chybová data na zařízení, ignorují chybové kódy přicházející zpět a zkouší použít zařízení, která neexistují nebo jsou aktuálně vytížená
    - Řízení zdrojů a vedení
        - Staré problémy se opět objevují, pokud programátor zakomponuje do programu nějakou starou verzi komponenty
    - Dokumentace - slabá dokumentace může způsobit ztrátu víry uživatele, že SW pracuje správně
    - Chyby testování
        - chybu způsobí špatně udělaný test
        - Jestliže program navádí většinu uživatelů ke způsobení chyb, pak je špatně navržen
- **Chyby požadavků, vlastností a funkčnosti**
    - Požadavky a specifikace
        - Neúplné, nejednoznačné, vzájemně si odporující
        - Hlavní zdroj drahých chyb
    - Chyby vlastností - chybějící, chybné, nevyžádané vlastnosti
- **Strukturální chyby**
    - Chyby v řízení sekvencí
        - Příkaz GOTO
        - Většina chyb řízení (v novém kódu) se dá snadno testovat a je chycena během testování jednotek
        - Neupravený starý kód může mít řadu chyb v řídicím toku
        - předčasná refaktorizace za účelem urychlení
    - Chyby logiky
        - Neporozumění jak se selekční či logické operátory chovají samostatně nebo v kombinacích
        - Neporozumění sémantice uspořádání logických výrazů a jeho vyhodnocení specifickými překladači
        - Chyby datového toku - nevztahují se k chybám řízení
        - Chyby toku řízení - část logického výrazu, která je použita pro ovládání toku řízení

- **Datové chyby**

    - Protože efekt poškození dynamických dat se může projevit velmi vzdáleně od příčiny, nalézají se takovéto chyby velmi obtížně
    - Základní problém zbytků ve sdílených zdrojích (např. vyčištění po použití uživatelem, sdílené čištění pomocí ovladače zdrojů, žádné čištění)

- **Chyby implementace**

    - Chyby kódování
        - Dobrý překladač chytne syntaktické chyby, nedeklarovaná data, procedury, kód a mnoho inicializačních problémů
        - Častou chybou kódu jsou dokumentační chyby (komentáře)
    - Chyby paměti

- Charakteristiky
  - Nejobtížnější chyby z hlediska lokalizace
  - Nejdůležitější chyby z hlediska opravy
  - Projevy nesprávného obsahu paměti jsou nepredikovatelné
  - Chyby v obsahu paměti se typicky projevují vzdáleně od jejich příčiny
  - Chyby zůstávají často nedetekované dokud nejsou náhodně spuštěny
- Typy chyb
  - Chyby hranic polí
  - Přístup přes nedefinovaný ukazatel
  - Čtení z neinicializované paměti
  - Chyby ztráty paměti (memory leaks)
- Slabá místa výkonnosti
  - Sběr správně vybraných dat
    - Řádka - kolikrát proběhla každá řádka - nejpřesnější, ale nejnáročnější na sběr dat
    - Funkce - méně podrobné než předchozí
    - Čas - data se sbírají z údajů časovaných běhů funkcí. Data jsou správná pro daný běh, ale závisí na stavu mikroprocesoru a paměti. Nejméně náročný sběr

3. Kritéria korektnosti a použitelnosti

- Obecné ukazatele kvality SW = Bugs per line of code, Code coverage, Cohesion, Coupling, Number of classes, LOC, operational complexity
- korektnost a použitelnost = intenzita defektů, bezpečnost proti útokům, použitelnost, udržovatelnost (jak snadno a rychle lze v nasazeném systému provádět změny)

4. Metriky kvality SW

- v různých vývojových fázích se uplatňují různé metriky (vývoj, testování, údržba, spokojenost zákazníka s nasazeným produktem atd.)

Střední doba k selhání (MTTF - mean time to failure)

- používá v bezpečnostně kritických systémech jakou jsou systémy řízení letového provozu
- implementačně velmi drahé

Intenzita defektů

- četnost defektů za určitý časový rámec
- ve jmenovateli vystupuje velikost softwaru (např. KLOC)

Efektivnost odstraňování defektů

DRE = (defekty odstraněné během vývoje / defekty setrvávající v produktu) * 100%

Řídící index nevyřízených věcí (BMI - backlog management index)

BMI = (počet problémů uzavřených během daného měsíce / přírůstek problémů během měsíce) * 100%

5. Modely spolehlivosti softwaru

Spolehlivost SW

= pravděpodobnost, že systém bude vykonávat svou funkci v daných operačních podmínkách po specifikovanou dobu
- používají se k odhadu spolehlivosti nebo počtu zbývajících defektů softwarového produktu, který byl uvolněn mezi zákazníky
- **důvody použití:**
  - objektivní vyjádření kvality produktu
  - plánování zdrojů pro fázi údržby SW

- sledovanou proměnnou studovaných kritérií je **počet defektů za daný časový interval** nebo **doba mezi dvěma selháními**.

a) Statický model

- parametry modelů jsou odhadovány na základě řady předchozích projektů

b) Dynamický model

- používá **průběžného vývoje vzorů defektů** k odhadu spolehlivosti finálního produktu
- založen na **exponenciálním růstu spolehlivosti** - vrchol přírůstků defektů je předpokládán na počátku, poté klesá (spolehlivost SW roste s časem)
- pro odhady se používají data z fáze formálního testování (jsou vhodným indikátorem spolehlivosti produktu pociťovanou zákazníky)

Základní předpoklady

- čím více defektů je objeveno a odstraněno dříve, tím méně jich zůstane na pozdější fáze
- jestliže každý krok vývojového procesu se provede s minimálním vznikem chyb, pak finální produkt bude dobrý
- vzniklé chyby se mají odstraňovat co nejdříve

Spolehlivost a validace predikce

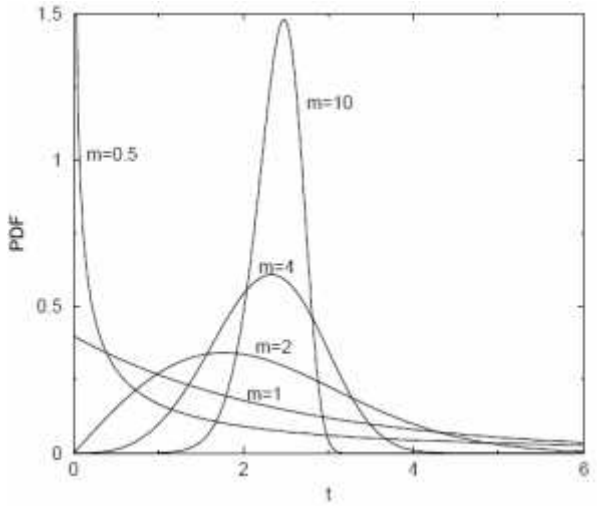
- **spolehlivost** vyjadřuje stupeň změny výstupu modelu vzhledem k možnostem fluktuací ve vstupních datech
- větší vzorky přesných a reálných dat lépe vypovídají o naměřených datech
- používat **více modelů** a spoléhat se na jejich společné hodnocení
- **platnost** se hodnotí porovnáním odhadů z modelů a jejich skutečných hodnot

Weibullova distribuce

- konce hustoty pravděpodobnosti se blíží asymptoticky k nule, ale nikdy ji nedosáhnou
- **Kumulativní distribuční funkce** (CDF):

$$F(t) = 1 - e^{-(t/c)^m}$$

- funkce hustoty pravděpodobnosti (PDF) znamená hustotu defektů v době přírůstkového vzoru defektů a CDF znamená kumulativní vzor přírůstku defektů



- **exponenciální model** (m=2) a **Rayleighův model** (m=1) jsou speciálním případem

Model Jelinski-Moranda (J-M)

- model doby mezi selháním
- předpoklady:
  - SW má N nedostatků na počátku testování
  - Selhání se projeví čistě náhodně
  - Všechny vady přispívají rovnocenně k příčině selhání během testování
  - Čas opravy je zanedbatelný
  - Oprava defektu je dokonalá
- doba mezi selháním (i - 1) a i, je dána **hazardní funkcí** v čase t_i:
- jak se jednotlivé vady odstraňují, doba do dalšího selhání bude delší

$$Z(t_i) = \phi[N - (i-1)]$$

ϕ je konstanta úměrnosti.

Modely Littlewooda

- podobné předchozímu Jelinsky-Moranda
- předpokládá se, že **různé vady mají různou pravděpodobnost, že povedou k selhání**
- vady většího rozsahu mají tendenci být detekovány a opraveny dříve
- zohledněním velikosti chyby se více blíží realitě

Goel-Okumotův model

- počítá s nedokonalou opravou a nenulovým časem opravy