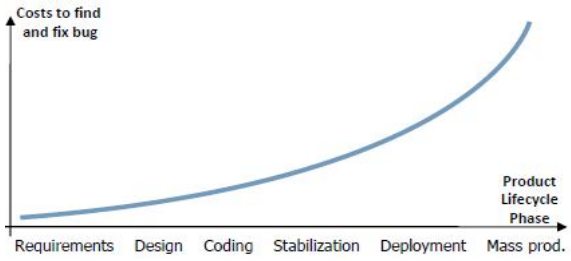
79. Jaký je smysl testování? Vysvětlete následující tvrzení: testování snižuje náklady.

Testování = spouštění programu se záměrem najít v něm defekty (tj. snažíme se, aby se projevily symptomy případných defektů)

- ověřuje, jestli je SW produkt správný (proti specifikaci)

Smysl:



- Kvalita
- Funkčnost
- Použitelnost
- Stabilita
- Bezpečnost
- Výkon
- Kompatibilita
- Perfektní vývojový proces není možný. Není možné zaručit bezchybnost SW.
- Testování snižuje náklady - V průběhu životního cyklu produktu náklady na nalezení a opravení chyby exponenciálně rostou (viz obr.)

80. Vysvětlete rozdíl mezi testováním nových funkcí a stabilizací?

Testování nových funkcí - snažíme se testovat novou funkci tak, aby obsáhla všechny možné varianty - snažíme se odhalit chyby

Stabilizace - nad samotným softwarem aplikujeme operace, které zvýší jeho stabilitu, tedy sníží riziko neočekávaného chování

- oprava chyb
- optimalizace, vylepšení
- cílem je tento čas co nejvíce snížit (1-2 měsíce)
- řešení chyb a jejich ověření
- zkušební protokol
- systémové testování - instalace a základní testy na všech podporovaných platformách
- interní testování - zpětná vazba z reálného prostředí, alfa tetování
- testování partnerů - beta testování, testování v praxi

81. Vysvětlete pojmy black box a white-box testování.

= dva základní přístupy k testování

Black box testování

- tester na program pohlíží jako na černou skříňku s danou specifikací, vnitřní struktura a vnitřní funkce programu ho nezajímají

- hledá případy, ve kterých se program nechová podle specifikace

- pro nalezení všech defektů by bylo nutné otestovat program se všemi možnými vstupy (platnými i neplatnými), což je prakticky nemožné (např. překladač jazyka C bychom museli otestovat se všemi platnými i neplatnými programy)

- Otázka: jak maximalizovat počet defektů nalezený konečným počtem testovacích případů? -> k programu nemůžeme přistupovat čistě jako k černé skříňce, ale musíme učinit rozumné předpoklady o jeho vnitřním chování

White box testování

- pro úplné otestování programu bychom potřebovali pomocí testovacích případů otestovat všechny možné logické cesty v programu (analogie otestování programu se všemi možnými vstupy, viz výše)

- ale - dva zásadní problémy:

1. počet logických cest je i v malých programech příliš velký – například for do 100 a v něm if = 2^{100} cest

2. i po otestování všech logických cest mohou v programu zůstat nenalezené defekty - některé logické cesty mohou chybět a nemusejí být nalezeny defekty citlivé na data, např. if ((a - b) < epsilon) ... // místo: if (abs(a - b) < epsilon)

- při black-box i white-box testování se budou testovací případy skládat z popisu vstupních dat a z popisu správného výstupu pro daná vstupní data - program nebo jeho část spustíme se vstupními daty, porovnáme předpoklad se skutečným výstupem

- testovací případy mají obsahovat platné i neplatné vstupy

- testovací případy je třeba uchovávat, protože je můžete znovu potřebovat (např. pro otestování programu po změně)

- je nutné také zkontrolovat, zda program neprovádí nechtěné vedlejší efekty (zápisy do databáze apod.)

82. Vysvětlete pojem akceptační testování a regresní testy.

Akceptační testování - test, zda produkt splňuje požadavky zadavatele

- testuje zadání na reálných datech, testuje pouze externí chování systému, vnitřní strukturu přitom ignorujeme

- obsah testu má specifikovat zadavatel (testy na straně zákazníka, v jeho prostředí)

- obsahem by měly být instance případů použití (tj. obsah funkčního testu je výhodné specifikovat v souvislosti se sběrem požadavků)

- snaha o zautomatizování, abychom mohli spouštět po změnách aplikace

Regresní testy se využívají při opětovném testování funkcí a vlastností aplikace (retesty opravených chyb, nový release)

- smyslem je ověření, že provedené změny či implementace nových vlastností v aplikaci nemělo žádný vliv na stávající funkce

- především na oblasti, které zůstaly v programovém kódu nezměněny

- oblasti, které byly předmětem úprav, by správně již měly být otestovány funkčními testy

- vhodné autotesty

83. Jak byste otestovali cyklus s podmínkou na začátku?

Musíme vytvořit tolik testovacích případů, aby se v každém příkazu "if" vykonala alespoň jednou větev při podmínce "false" a alespoň jednou větev při podmínce "true" atd.

Rozumným kritériem je pokrytí všech kombinací podmínek v rozhodovacím příkazu (multiple condition coverage).

84. Jak byste otestovali validitu e-mailové adresy?

Regulárním výrazem.

85. Vysvětlete pojmy testování jednotek, JUnit.

Testování jednotek

- "jednotka" = procedura, funkce, nebo nejmenší samostatně přeložitelná jednotka zdrojového textu
- používají se white-box techniky
- pro objektově-orientovaný software se za jednotku považuje třída, testování třídy:
    - samostatné otestování každé metody
        - některé metody lze testovat až po předchozím vyvolání jiných metod, např. po inicializaci objektu
        - pokud používáme dědičnost, musíme testovat i všechny zděděné operace
    - testovat průchod všemi stavy objektu, příp. simulace všech událostí, které způsobují změnu stavu objektu
        - pokud jsme vytvořili stavový diagram objektu, můžeme z něj určit posloupnost přechodů, které chceme testovat, a najít posloupnost událostí, které ji způsobí
    - testovat nastavení všech atributů objektu

Nástroj JUnit

- knihovna JUnit pomůcka pro testování jednotek v jazyce Java
- poskytuje nástroje pro spouštění testovacích případů a umí graficky i textově zobrazit výsledky testů
- pro jednotkové testy je důležité mít také zavedené konvence, například:
    - pro každou třídu "Třída" vytvoříme třídu "TřídaTest"
    - pro každou veřejnou metodu vytvoříme testovací metodu "testJménoMetody()"
    - testujeme vždy správné chování i chování při chybách, testovací metody můžeme někdy rozdělit do 2 skupin:
        - testJménoMetody - testuje obvyklé chování
        - testJménoMetodyFailures - testuje chování při chybách

86. Vysvětlete pojmy integrační testování, validační testování.

Integrační testování = sestavujeme SW, spolu s tím testujeme defekty týkající se rozhraní mezi jednotlivými částmi

- po otestování individuálních komponent musíme komponenty integrovat - sestavit do částečného nebo úplného systému
- výsledek musíme otestovat na problémy, které vznikají interakcí komponent
- (velký třesk) - po otestování jednotlivých modulů je z nich v jediném kroku sestavena aplikace
- použitelné pro malé programy, pro větší systémy nejméně efektivní způsob integrace = vysoká pravděpodobnost neúspěchu
- hlavním problémem je lokalizace defektů, protože vztahy mezi komponentami mohou být značně složité
    - proto často pro integraci a testování doporučuje inkrementální přístup
    - nejprve integrujeme minimální konfiguraci systému, otestujeme
    - k systému přidáváme inkrementy, po každém přidání systém otestujeme
    - pokud nastaly problémy, budou pravděpodobně (ale ne nutně) způsobeny přidáním posledního inkrementu
- pokud má důležitý modul neočekávané problémy, může se tím zdržet celá integrace (programátor řeší problém, zatímco všichni ostatní na něj čekají)

Validační testování

= vyhodnocení SW na konci procesu vývoje SW, abychom zajistili splnění požadavků na SW

- odpověď na otázku: vytvářím správný produkt?
- začíná tam, kde končí integrační testování
- testujeme, zda SW splňuje požadavky uživatele, se zaměřujeme na funkce viditelné uživatelem
- akceptační testování - test zda produkt splňuje požadavky zadavatele, testuje zadání na reálných datech, testuje pouze externí chování systému, vnitřní strukturu přitom ignorujeme
    - obsah testu má specifikovat zadavatel
    - obsahem by měly být instance případů použití (tj. obsah funkčního testu je výhodné specifikovat v souvislosti se sběrem požadavků)
    - snaha o zautomatizování, abychom mohli spouštět po změnách aplikace

87. Vysvětlete pojmy alfa a beta testování, zátěžové testování.

Alfa a beta testování

- pro generické produkty není většinou možné vykonat přejímací testování u každého zákazníka, proto alfa a beta testování
- alfa testy: na pracovišti, kde se SW vyvíjí (známé prostředí) - testuje uživatel, vývojoví pracovníci ho sledují a zaznamenávají problémy
- beta testy: testují vybraní uživatelé ve svém prostředí (vývojářům neznámém) - defekty ohlášené uživateli jsou opraveny => finální produkt

Zátěžové testování

- zátěž = množství dat, frekvence požadavků, data, která jsou extrémně náročná na zpracování
- obvykle se používají testy, kde se zátěž postupně zvyšuje, dokud není výkonnost systému neakceptovatelná nebo dokud systém nehavaruje
- je vhodné určit části kódu, které mohou být problematické při velké zátěži, zátěžové testy navrhnout tak, aby pokrývaly především tyto části kódu
- ověření, zda havárie systému nepoškodí data apod.
- může odhalit některé defekty, které se normálně neprojeví
- důležité zejména u internetových aplikací, v distribuovaných systémech apod., kde se vysoce zatížené systémy mohou zahltit, protože si vyměňují mnoho koordinačních dat, čímž se opět zvyšuje zátěž systému atd.

88. Co jsou to autotesty, jaký je jejich význam, jaké mají výhody a nevýhody?

= automatizace testů, automatické systémy, nástroj pro testování SW podle stanovených kritérií

- skripty simulují práci testera

- rozsáhlé systémy - cena testování až 50% ceny vývoje, mohou existovat stovky až tisíce testovacích případů

- mnoho podporovaných prostředí - OS, prohlížeče

Výhody:

- automatizace testů snižuje cenu změn - programátoři se nemusejí tolik obávat, že změnou zanesou do kódu defekt, protože testy by defekt (s určitou pravděpodobností) odhalily
- rychlost, efektivita testování
- redukuje čas potřebný pro testování
- pracuje přesně, nedělá chyby a pracovníci se mohou věnovat jiné práci

Nevýhody:

- nelze otestovat vše
- tvorba skriptů a údržba - chyby v kódu, zvyšují se náklady
- testy najdou obvykle méně než 60% defektů, proto se kombinují s inspekcemi
- defekty jsou častěji v testovacích případech než v testovaném kódu

89. Jak v závislosti na stavu výsledků testů poznáte, že můžete vydat produkt?

V případě dobrých výsledků testů se může produkt vydat. Výsledky testů nemusí být 100%. Záleží na druhu produktu a cílové skupině, pro kterou je produkt určen.

Test by měl být 100% v námi známé a otestované oblasti. Neměli bychom vydat produkt, kde známe chybu, ale neopravili jsme ji. Samozřejmě žádný produkt nemůže být 100% funkční, ale vše známé by mělo být odstraněné.

90. Vysvětlete podstatu následujících procesů v podniku: systémová integrace, provoz ICT, servis ICT

5 procesů:

Máme-li informační systém, tak musíme pro jeho dlouhodobé efektivní fungování umět zajistit

- Systémová integrace představuje zavedení SW a HW ve firmě

- Provoz ICT - cílem je zajistit IS a infrastrukturu dostupnou, připravenou pro použití a funkční

- Provoz ICT se neobejde bez servisu ICT

91. Vysvětlete pojem dostupnost, lhůta plnění, základní doba služeb v oblasti provozu IS.

- dostupnost - doba, kdy je garantována funkčnost IS – např. 95%

- lhůta plnění - pro vykonávání stanovených činností - do kdy musí být nahozen server po spadnutí

- základní doba služeb - doba, ve které jsou garance a lhůty plnění poskytovány - od 7 do 7 třeba

92. Vysvětlete princip činnosti primární a sekundární podpory.

Primární podpora

- kontaktní místo pro pomoc uživatelům (Hot-line, HelpDesk)
- poskytování řešení známých problémů, používá známé postupy (odpověď obratem)
- sledování provozu
- poskytuje informace

Sekundární podpora

- poskytuje řešení problémů
- řešení nestandardních provozních úloh
- řešení drobných změn
- statistiky
- poskytuje řešení u předem neznámých nebo technologicky vázaných požadavků a problémů

93. Jaké je základní pravidlo při řešení problémů servisu informačních systémů?

Základní pravidlo: Nahlášení problému

3 kategorie podle závažnosti a severity:

- A – kritický problém, nefunkčnost, kterou nelze nijak obejít
    - vyžaduje okamžité řešení, Urgent severita
    - buď odstranění problému, nebo dočasné náhradní řešení (uživatel může pokr. v práci) a poté systémové řešení
- B – vážný problém, který lze obejít náhradním řešením
- C – problém, který nemá dopad na funkčnost

94. Jak vypadá SLA v oblasti servisu IS?

Service level agreement

- je smluvní dohodou mezi dodavatelem a odběratelem
- definuje úroveň a kvalitu služeb jako parametry
- pro porušení parametrů stanoveny sankce
- stanovuje cenu rizika
- lhůty plnění (lhůta pro zahájení řešení problému, lhůta pro snížení kategorie problému, lhůta pro odstranění problému)
- základní doba služeb

95. Vysvětlete pojmy provozní, testovací a vývojové prostředí v oblasti servisu IS.

96. Jaké právní úpravy platí v oblasti softwarového práva?

Zákon č. 121/2000 Sb., autorský zákon

Zákon č. 441/2003 Sb., o ochranných známkách

Zákon č. 365/2000 Sb., o informačních systémech veřejné správy

Zákon č. 127/2005 Sb., o elektronických komunikacích

Obchodní,občanský a trestní zákoník, zák. o ochraně osobních údajů

97. Vysvětlete pojem autorské dílo, uveďte příklady autorských děl, co není autorským dílem?

Autorským dílem je jedinečný výsledek tvůrčí činnosti autora, dílo v objektivně vnímatelné podobě, ©

- dílem je hudba, počítačový program, grafické, choreografické, fotografické, audiovizuální, architektonické dílo
- dílem není námět, zpráva, informace, metoda, teorie, vzorec, graf, tabulka fyzikálních konstant, výstup počítačového programu

98. Popište princip dualismu autorského práva – právo majetkové a právo osobnostní.

Autor = programátor nebo člen tvůrčího týmu

- Osobnostní autorská práva jsou spojena pouze s osobou autora a smrtí autora zanikají
  - Právo osobovat si autorství
  - Právo na zveřejnění
  - Právo na nedotknutelnost
- Majetková autorská práva s ekonomickým významem (Trvají ještě 70 let po smrti autora)
  - Právo užít SW nebo udělit svolení k užívání
    - Právo rozmnožování
    - Rozšiřování rozmnoženiny
    - Pronájem nebo půjčování
    - Sdělování veřejnosti
    - Odměna: Opětovný prodej originálu, rozmnoženiny pro vlastní potřebu

99. Popište základní typy licencování softwaru a základní obsah licenční smlouvy.

- patří do softwarového práva

Licence = právo na různé způsoby užití díla je možno převádět, prodávat apod.

2 základní typy:

- výhradní - poskytovatel není oprávněn licencovat 3. osobě nebo užívat sám
- nevýhradní - neomezuje poskytovatele v dalších dispozicích se SW

Licenční smlouva

- specifikace SW
- právo a způsob užití SW, rozsah licence
- odměna za licenci

100. Je dán následující scénář případu užití:

- ***Klient vloží kartu. Bankomat kartu přečte a zjistí její sériové číslo.***
- ***Bankomat požádá uživatele o zadání PIN; uživatel zadá "1234".***
- ***Bankomat ověří číslo karty a PIN u centrálního počítače.***
- ***Bankomat požádá o zadání velikosti částky; uživatel zadá 1000 Kč***
- ***Bankomat požádá centrální počítač o provedení transakce; centrální počítač transakci provede a vrátí nový zůstatek účtu.***
- ***Bankomat vydá částku, vytiskne stvrzenku a vrátí kartu.***

Nakreslete odpovídající sekvenční diagram.