Strategije testiranja

- Programska oprema je testirana z namenom odkrivanja napak, ki so bile storjene pri načrtovanju ali izdelavi
- Kako testirati?
  - Formalen načrt testiranja
  - Testiranje celotne aplikacije naenkrat
  - Testiranje posameznih komponent
  - Ponovno testiranje po dodajanju novih komponent
  - Vključevanje uporabnikov v proces testiranja

- Izdelava strategije testiranja: projektni vodja, sistemski inženirji, specialisti za testiranje
- Zakaj je izdelava strategije testiranja pomembno?
  - Za testiranje porabimo več časa kot za katerokoli drugo aktivnost
  - Če se testiranje izvaja adhoc, je to izguba časa, potrebno je še več napora in določene napake ostanejo neodkrite

  Zato je smiselno izdelati sistematično strategijo testiranja!
- Kakšen je postopek testiranja?
  - Začne se pri testiranju posameznih komponent
  - Nadaljuje pa s testiranjem integracije večih ali vseh komponent v sistem
  - Odkrite napake moramo popraviti s razhroščevanjem

- Specifikacija strategije testiranja je dokument, ki specificira pristop ekipe razvijalcev k problemu testiranja; od splošne strategije do posameznih korakov testiranja
- Pregled specifikacije strategije testiranja:
  - Zagotavljanje kompletnosti
  - Pregled posameznih nalog testiranja
- Različni vzorci strategij:
  - Formalne tehnične revizije
  - Ni nujno, da je en način testiranja primeren tekom celotnega razvoja
  - Testiranje izvajajo razvijalci ali neodvisna testna ekipa
  - Razhroščevanje mora biti del vsak strategije testiranja

Konflikt interesov

- Ljudje, ki razvijejo programsko opremo so v fazi testiranja zadolženi za testiranje njihovega dela.
- Vsi razvijalci imajo interes dokazati:
  - da je njihova programska oprema brezhibna,
  - da ustreza zahtevam naročnikov,
  - da bo razvita v roku in v okviru projektnega proračuna.
- Testiranje je proces, ki poskuša dokazati nasprotno.
- Nevarnost: načrtovanje testov, ki bodo dokazali brezhibnost PO, namesto da bi pomagali odkrivati napake
- Napake bodo vedno prisotne!

Neodvisna testna ekipa

- Odpravlja probleme, ki se pojavljajo s konfliktom interesov
- Plačani so za to, da najdejo napake
- To ne pomeni, da razvijalci dajo svojo programsko opremo NTE in je za njih delo končano
- Razvijalci in NTE morajo tesno sodelovati in zagotoviti, da so vsi testi korektno izvedeni
- Razvijalci morajo biti pripravljeni popraviti odkrite napake

Vrste testiranja

- Unit testi
- Testi integracije
- Validacijski testi
- Sistemski testi

Kdaj smo opravili dovolj testiranja?

Statistični odgovor: kadar lahko s 95% gotovostjo trdimo, da je verjetnost, da v 1000 urah delovanja programa ne odkrijemo niti ene napake, vsaj 0.995.

PO mora biti narejena tako, da olajša testiranje (avtodiagnoza).

Unit testi

- Opravljeni na najmanjših enotah PO – komponenti ali modulu
- Testira se:
  - vmesnik komponente,
  - lokalne podatkovne strukture,
  - robne pogoje,
  - možne poti skozi metode,
  - kodo za upravljanje z napakami.
- Okolje za unit teste: upravljalec (driver), nadomestki (stub).
- Poenostavljeno je testiranje komponent z visoko stopnjo kohezije in tistih, ki opravljajo eno samo bistveno operacijo.

Testi integracije

- Če vsaka komponenta posebej deluje pravilno, zakaj ne bi deloval pravilno tudi skupek komponent?
  - podatki se lahko izgubijo pri prenosu prek vmesnikov,
  - ena komponenta ima lahko nepredviden efekt na drugo,
  - zaporedje izvajanja metod lahko ne da želenih rezultatov,
  - računske napake se lahko povečajo prek sprejemljivih okvirov
  - globalne spremenljivke lahko predstavljajo problem

Testi integracije

- Integracija je sistematična tehnika za izgradnjo arhitekture programske opreme, kjer hkrati izvajamo teste za odkrivanje napak pri kombiniranju različnih komponent
- Big bang pristop
- Inkrementalna integracija:
  - Z vrha navzdol (glavni modul/komponenta, nadomestitev enega nadomestka z realno komponento, testiranje, regresijsko testiranje)
  - Od dna proti vrhu (združitev komponent na najnižjem nivoju v grozde, kreiranje upravljalca grozda, testiranje grozda, odprava upravljalca in združevanje grozdov)

Regresijski testi

- Ponovno izvajanje določene podmnožice testov ob integraciji nove komponente
- Zagotavljanje, da nova komponenta nima nezaželenih stranskih učinkov
- Podmnožica testov vsebuje:
  - Reprezentativen vzorec testov, ki preveri delovanje vseh funkcionalnosti PO
  - Dodatne teste funkcionalnosti, za katere obstaja največja verjetnost, da jih nova komponenta prizadene
  - Teste komponent, ki so bile spremenjene

Validacijski testi

- Izvajajo se potem, ko so vse komponente integrirane v sistem
- Validacija je uspešna, ko se PO obnaša tako kot pričakuje uporabnik
- Validacijski testi morajo zagotoviti preverjanje obnašanja PO v skladu s funkcionalnimi zahtevami
- Kadar je PO razvita za enega naročnika, so validacijski testi lahko kar testi sprejemljivosti PO, ki jih izvede naročnik
- Kadar je PO razvita za veliko uporabnikov, lahko izvedemo dve vrsti validacijskih testov: alfa teste in beta teste.

Sistemski testi

- Integracijski in validacijski testi s realno strojno opremo, ljudmi in informacijami
- Upravljanje s potencialnimi napakami pri prenosu informacij med različnimi deli sistema:
  - testi, ki preverijo informacije, pridobljene iz drugih delov sistema
  - testi, ki simulirajo pošiljanje neustreznih podatkov
  - zapis rezultatov takih testov

Sistemski testi

- Pobiranje po napakah
- Varnostni testi
- Stres testi
- Performančni testi

Razhroščevanje

- Posledica uspešnega testiranja
- Razhroščevanje je proces odkrivanja vzrokov in odpravljanja napake, ki jo odkrijemo s testiranjem
- Lahko se zgodi, da manifestacija napake (simptom) v delovanju programa in vzrok zanjo nimata nobene očitne povezave
- Napake imajo lahko različno težo: od rahlo nadležnih (npr. napačen format izpisa) do katastrofalnih (sistemske napake, ki povzročijo resno ekonomsko ali fizično škodo)

Karakteristike napak v PO

- Simptom in vzrok napake sta lahko “geografsko” oddaljena
- Simptom lahko (začasno) izgine, ko je popravljena neka druga napaka
- Simptom se lahko pojavi zaradi nenapak (npr. nenatančnost pri zaokroževanju realnih števil, ipd.)
- Simptom je lahko posledica človeške napake pri uporabi PO
- Simptom je lahko posledica časovnih težav
- Včasih je zelo težko ponoviti pogoje, ki so pripeljali do simptoma
- Simptom se lahko pojavlja v časovnih intervalih
- Simptom je lahko posledica večih vzrokov, ki so razpršeni po različnih procesih, ki tečejo na različnih procesorjih

Tehnike razhroščevanja

- Metoda grobe sile
- Retro analiza
- Eliminacija vzrokov
- Avtomatsko razhroščevanje
- Drugi programerji

Odpravljanje napak

- Odpravljanje napake lahko povzroči nove
- Preden se lotimo odpravljanja napake, je treba odgovoriti na naslednja vprašanja:
  - ali se vzrok napake pojavlja še v kakšnem delu programa?
  - kakšna nova napaka bi se lahko pojavila pri odpravljanju trenutne?
  - kaj bi lahko naredili, da do te napake sploh ne bi prišlo?