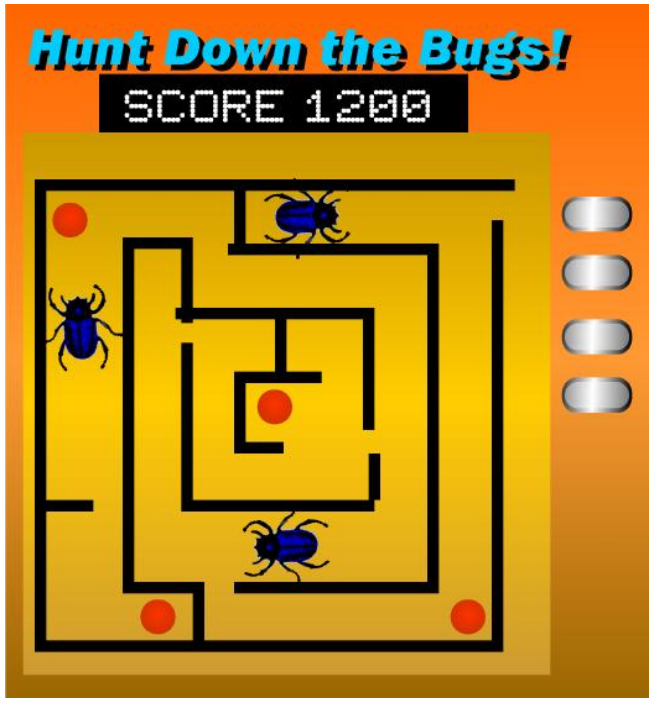
Tehokas vianetsintä – taktiikoita testaajille



Joukko erilaisia periaatteita ja taktiikoita, jotka antavat lisätehoa ohjelmiston vikojen löytämiseen. Periaatteita voi soveltaa sekä testien systemaattisessa etukäteissuunnittelussa että ketterässä testauksessa.

UPDATE 2010

Matti Vuori, www.mattivuori.net

Sisällysluettelo 1/2

Sisällysluettelo 2/2

Asenne

- Testauksessa on tavoitteena löytää vikoja, jos niitä ei löydy, testaus ei ole onnistunut.
- Ohjelmistossa on vikoja – ne vain ovat vielä löytämättä.
- Ohjelmiston pitää kestää kovakouraistakin kohtelua testauksessa – muuten se ei ole riittävän tukeva tuotantokäyttöönkään.
  - Ohjelmisto voi kokea käytön aikana mitä tahansa.
  - Suhtaudu ohjelmistoon armottomasti ja ennakkoluulottomasti.
- Mieti, että nopeassa testausjaksossa olisi löydettävä kaikki ne viat, jotka muuten paljastuisivat vähitellen suuren käyttäjäjoukon kenties vuosien työn kuluessa.
  - Ohjelmistoa on siksi todella koeteltava testauksessa.
- Testaus perustuu raadolliseen todellisuuteen, sen tehtävä on purkaa pilvilinnat

Varoituksen sana

- Tässä kalvosarjassa puhutaan vain siitä, millaisilla ajattelumalleilla bugeja löydetään
- Aivan toinen asia ja pitkien tarinoiden aihe on se, miten testausta suunnataan sen prioriteettien ja riskien perusteella. **Joka paikasta löydetyt virheet eivät ole yhtä tärkeitä!**
- Kuitenkin: määrässä on laatua! Tiettyä määrää pieniäkin bugeja vastaa vähäisempi määrä isompia bugeja. **Pienten bugien löytämisellä päästään isojen jäljille!**



Ymmärrä koko ohjelma 1/2

- Ymmärrä, miten koko ohjelma toimii.
- Vaikka tehtävänä olisi testata vain pientä osaa ohjelmasta, on tärkeää tuntea sen osuus kokonaisuudesta
  - Mitä koko ohjelma tekee
  - Miten sitä käytetään, kuka, millä tavoilla ja tavoitteilla, millaisilla poikeamilla
  - Miten kukin palanen liittyy kokonaisuuteen
  - Mistä tiedot tulevat, mihin ne menevät, miten ne virtaavat
  - Mikä on tärkeää ja mikä ei

Ymmärrä koko ohjelma 2/2

- Joskus vaatimusmäärittelyjen ja testauksenhallinnan näkymät sekä tehtävänantosi vähentävät ymmärrystä esittelemällä kokonaisuudesta vain siivuja – yritä nähdä enemmän

Testaa koko tuote, kaikki käyttö

- Testaa järjestelmällisesti kaikki toiminnot ja ominaisuudet.
- Suorita kaikki ohjelmiston elinkaaren, käyttötapausten ja toimintoketjujen tapahtumat.

Tiedä kaikki vaatimukset

- Käytä kaikkia tietolähteitä ohjelmiston vaatimusten tunnistamiseen.
  - Määrittelyt, käyttöohjeet, tieto käyttäjien tavoista, yleinen tietämys siitä miten tällaisia tuotteita käytetään.
- Muista, että määrittelypapereissa on vain pieni osa vaatimuksista.
- Mieti, mitä käyttäjät odottavat.
- Selvitä, mitkä asiat ovat ohjelmiston käytössä tärkeitä.

Etsi kaikenlaisia virheitä

- Puuttuva toiminto.
- Väärin toimiva toiminto.
- Hankalakäyttöisyys.
- Käyttö- tai tietoturvallisuusongelmat.
- Suorituskykypuutteet.
- Määrittelypuutteet – määrittelyn mukaan toteutettu toiminto toimii huonosti tai epäloogisesti.
- Esteettiset puutteet (on toki ymmärrettävä, että omat preferenssit voivat olla erilaisia kuin ohjelmiston kohderyhmällä).
- Tunnista tyypillisiä tai mahdollisia vikoja.
  - Preppaa itsesi ennen testausta tarkistuslistojen ja aiempien ohjelmistojen virheiden avulla.

Ole realisti

- Testaa sellaisilla tavoilla, joilla ohjelmistoa käytetään oikeasti.
- Muista, että käyttäjät eivät tiedä yhtään mitään siitä, miten koodaajat ovat ajatelleet ohjelmaa käytettävän "oikein".
- Ajattele kuin käyttäjä – kuin aloittelija, kuin hyvin kokenut, monipuolinen ja vaativa experttikäyttäjä.
- Käytä testattavia toimintoja useilla erilaisilla tavoilla.
- Käytä huonosti käyttäytyvää testiaineistoa (puuttuvia tietoja, väärämuotoista, jne…)

Rakasta virheitä

- Kun jotain vihaa, ei osaa ajatella selkeästi ja luovasti
- Siksi ohjelmiston virheitä ei kannata vihata, vaan rakastaa!
- Jokainen löydetty virhe kertoo testauksen onnistumisesta ja on oppimisen mahdollisuus koko organisaatiolle – teemalla: miten sellainen virhe vältetään jatkossa?

Kyseenalaista vanha testaustapa.

- Uudet viat eivät välttämättä löydy vanhoilla testaustavoilla ja testeillä.
- Uudista testejä koko ajan.
- Suunnittele ja suorita sellaisia testitapauksia, joita ei ole aikaisemmin käytetty.

Käytä erilaisia testiympäristöjä

- Eri käyttöjärjestelmäversiot.
- Erilaiset tietokoneet.
- Erilaiset oheisohjelmat.
- Jokainen ympäristön muutos auttaa bugien löytämisessä.
- Pyri matkimaan erilaisten käyttäjien ympäristöjä.
  - Realismi ja ”likaisuus” (paljon epämääräistä softaa, pitkä historia) auttaa paljastamaan bugeja
- Älä käytä samanlaista ympäristöä kuin ohjelmistokehittäjillä – käyttäjillä ei ole sellaista ympäristöä.
- Mieti käyttöympäristön mahdollisia ongelmia. Toimiiko verkko aina ok? Vedä verkkojohto seinästä!

Konfiguroi ympäristöt eri tavoilla

- Esimerkiksi nettipalvelimet voivat olla tuhansilla tavoilla konfiguroituja – ja ovatkin, käytännössä
- Kokeile muuttaa asetuksia
- Ruuvaa tietoturva-asetukset kaikkein tiukimmalle – niin tulee joku käyttäjäorganisaatio tekemään

Asetu tietoisesti rooleihin

- Aseta itsesi erilaisiin käyttäjärooleihin.
- Toimi kuin
  - Aloittelija.
  - Lapsi.
  - Vanhus.
  - Huippexpertti.
  - Nörtti.
  - Vieraskielinen.
- Mitä he tekevät? Toimi niin. Millaisia virheitä he tekevät? Tee niitä.

Tutki poikkeus- ja virhetilanteita

- Testaa poikkeustilanteita ja raja-arvoja.
  - Kokeile oudoilla virhesyötteillä – niin käyttäjätkin tekevät.
  - Kokeile raja-arvoilla ja arvoalueen ulkopuolella olevilla arvoilla – nolla, miinusmerkkiset luvut.
- Testaa virhetilanteita.
  - Yritä avata tiedosto, jota ei ole olemassa, virheellinen tiedosto, etsiä tietoa, jota ei ole olemassa.
  - Mitä tapahtuu, jos töpselin vetää seinästä?

Yllätä

- Tee odottamattomia asioita.
  - Käynnistä toiminto, jota ei "pitäisi" käyttää.
  - Aloita ohjelman käyttäminen sen latautuessa.
  - Tee asioita satunnaisesti.
  - Tee gorillatestausta.

Kuormita

- Kuormita ohjelmistoa.
  - Käytä sitä pieniresurssisella laitteistolla.
  - Avaa suuri määrä tiedostoja.
  - Käynnistä useita samanaikaisia toimintoja.
- Toista, toista, toista.
  - Monet viat ilmenevät vasta pidemmän käyttöjakson myötä.
  - Toiminnot voivat toimia ensimmäisellä kerralla, mutta eivät enää seuraavalla.

Käytä vainuasi

- Haasta koodaaja. Mieti, mitä asioita hän ei ole kenties ottanut huomioon.
- Luota "metsästäjän vainuun".

Tunne vikojen luonne

- Viat pitävät toisistaan.
  - Jos jollakin osa-alueella on paljon vikoja, niitä voi löytyä paljon lisää!
- Testaa piilossa olevia toimintoja.
  - Päänäytön toiminnot luultavasti toimivat, mutta pari tasoa siitä alaspäin, vikoja voi olla enemmänkin.

Tiedosta ohjelmiston toteutuksen haasteet

- Tunne ohjelmiston toteutuksen kriittiset piirteet.
  - Jos ymmärrät tekniikan päälle, testaa sellaisia asioita, joiden tiedät olevan vaikeaa toteuttaa luotettavasti.
- Testaa muuttuvia alueita.
  - Muutos on aina mahdollisuus vioiksi.
  - Aina menee jotain rikki – löydä se.
  - Testaa alueita, jotka ovat uusia, joihin on aiemmin kohdistunut monia muutospyyntöjä.
- Testaa uuden teknologian sovellukset huolella
- Myöhässä olevat osat tehdään paineissa ja ovat helposti täynnä vikoja

Selvitä vaikea toteutuksen paikat

- Juttele ohjelmistokehittäjien kanssa ja selvitä,
  - Minkä asioiden toteutuksessa on ollut vaikeaa
  - Mitkä paikat ohjelmassa pysyvät kasassa vain hyvällä onnella…
  - Missä asioissa kehittäjät tietävät tulevan ongelmia

Monimutkaisuus tuottaa virheitä

- Kun ohjelman logiikka on monimutkaista, bugit ovat aina lähellä.
  - Erityisesti silloin, kun logiikka vaatii substanssitietoa
- Jos et saa selvää dokumenteista, eivät muutkaan ole saaneet – joten väärinkäsityksistä on syntynyt virheitä
- Jos johonkin toimintoon liittyy paljon dataa – kuten lomalleellinen nippelitietoa – osaa siitä käsitellään ohjelmassa yleensä väärin

Tunnista ongelmien oireita

- Jos jossakin toiminnossa esiintyy pieniä virheitä näytön päivityksessä, kursorin liikkeissä, hidastumista jne., siellä voi piillä isojakin vikoja.
- Tunnista puutteellista suunnittelua.
  - Alueilla, jotka ovat monimutkaisia, epäsiistejä, hitaita jne. on usein myös vikoja.
  - Testaa niitä alueita, joista on tullut helpdeskiin valituksia.

Katso ihmistä

- Uudet tiimit ja uudet työntekijät yllättävät usein – turhilla virheillä, ennen kuin pääsevät samalle viivalle muiden kanssa
- Koodaajat, joilla on aina kiire ja työpäivät venyvät, eivät välttämättä ehdi itse testata tuotoksiaan kunnolla
- (Koodaajien laaduntuotolla on selviä eroja, mutta usein on enemmän kyse olosuhteista, joissa he toimivat ja softan historiasta)

Hyödynnä kokemustietoa

- Testaa niitä alueita, joissa ohjelmistoissa on usein vikoja.
  - Mieti kokemustasi vastaavan tyyppisistä ohjelmistoista.
- Testaa aiempien versioiden vikojen kaltaisia asioita.
  - Viat syntyvät ohjelmistokehittäjien käsissä, samanlaisista suunnittelu- ja toteutustavoista.
  - Jos aiemmissa versioissa on ollut tulostusongelmia, testaa niitä tälläkin kertaa.
  - Hanki tietoa reklamaatioista ja havaituista virheistä (ylläpidon vikakanta, helpdesk-tietokanta).

Tee yhteistyötä ja ideoi

- Testaustiimin aivot kannattaa yhdistää vikojen potentiaalisten paikkojen tunnistamiseen.
- Erityinen eri osapuolten (testaajat, kehittäjät jne.) yhteinen "vianmetsästyspäivä" voi olla hyvin hedelmällinen.

Ajattele omilla aivoillasi

- Jokaisen ammatin ja työsuhteen alussa noudatetaan alan ja työpaikan käytäntöjä.
- Mutta joskus ne ovat kehnoja! … Ja varsinkin testauksessa, mikä on kovin uutta monille organisaatioille.
- Joskus organisaatiot ovat tapoihinsa kangistuneita.
- Siksi kannattaa tunnistaa oma, kehittyvä oppiminen ja huomata hetki, jolloin pystyy itse miettimään, miten testaus on hyödyllisintä tehdä.

Löydä oma intohimosi testaukseen

- Löydä se syy, miksi testaus on – tai voisi olla – hienoa.
- Rakenna tietoisesti mielenkiintoa eri asioihin ja anna mielenkiinnon kasvaa oppiessasi asioita.
- Näin saat motivoitunutta skarppiutta ja uutta bugien löytämisen kykyä!