Tampereen ammattikorkeakoulu

Ohjelmistotestaus

Santeri Puttonen

Raportti
Marraskuu 2024

Tietojenkäsittely

SISÄLLYS

1 JOHDANTO

Ohjelmistotestaus on keskeinen prosessi ohjelmointikehityksessä, jonka avulla voimme varmistaa, että ohjelmistot täyttävät niille asetetut vaatimukset ja toimivat halutulla tavalla kaikissa käyttötilanteissa.

Ohjelmistotestaus sisältää useita eri testausmenetelmiä ja tyyppejä, kuten staattinen ja dynaaminen testaus, sekä eri testauksen tasoja, kuten yksikkö- integraatio- ja järjestelmätestaus. Testauksen käsitteet, kuten varmistus ja validointi, ovat tärkeitä ohjelmiston laadun varmistamisessa.

Testauksen automatisointi on nykyään yhä tärkeämpää, sillä sen avulla voidaan tehostaa testausprosessia, parantaa sen kattavuutta ja vähentää inhimillisten virheiden mahdollisuutta ohjelmistoja testatessa. Automaattisen testauksen avulla voidaan suorittaa toistuvia ja monimutkaisempia testejä nopeasti ja tarkasti, joka säästää aikaa ja resursseja.

Tässä raportissa tarkastellaan ohjelmistotestauksen merkitystä, sen tavoitteita, eri testausmenetelmiä ja testaustyyppejä, sekä testauksen roolia ohjelmiston laadun varmistamisessa ja kehittämisessä. Lisäksi käsitellään testauksen automaation eri hyötyjä ja haasteita kuin myös sen laatutekijöitä ja -kriteereitä, jotka ovat kummatkin keskeisiä ohjelmistojen kehityksessä ja laadunhallinnassa.

2 Mitä on ohjelmistotestaus ja miksi sitä tarvitaan

Ohjelmistotestaus on prosessi, jonka avulla tarkistetaan, että ohjelmisto toimii halutulla tavalla. Testauksen avulla voidaan löytää mahdolliset virheet. Ennen ohjelmiston julkaisemista onkin syytä tehdä kyseinen testaus, sillä sen avulla voidaan varmistaa ohjelmiston toimivuus, turvallisuus sekä helppokäyttöisyys.

2.1 Ohjelmiston laadun merkitys

Ohjelmiston laatu on keskeinen tekijä, joka vaikuttaa merkittävästi siihen, miten hyvin ohjelmisto palvelee käyttäjiään. Testaus varmistaa, että ohjelmisto täyttää laatuvaatimukset ja toimii suunnitellulla tavalla kaikissa olosuhteissa. Esimerkiksi verkkopankkiohjelmisto, joka ei toimi luotettavasti, voi vaarantaa sen käyttäjien luottamuksen ja pahimmillaan vahingoittaa yrityksen mainetta.

Ohjelmistotestaus auttaa löytämään haavoittuvuuksia ja turvallisuusongelmia, jotka voivat altistaa ohjelmiston erilaisille hyökkäyksille. Turvallisuustesti auttaa tunnistamaan ohjelmiston heikkoudet, kuten haavoittuvuudet tai tiedonvuodot.

2.1.0 Testauksen tavoitteet

Ensisijaisena tavoitteena ohjelmistotestauksessa on virheiden löytö. Pienetkin virheet koodeissa voivat estää ohjelmiston oikeanlaisen toimimisen. Toinen tärkeä tavoite on toiminnallisuuden tarkistaminen. Toiminnallisuuden avulla varmistetaan, että ohjelmisto toimii oikein ja täyttää sille vaaditut liikennetoimintavaatimukset. Tällä tarkoitetaan sitä, että ohjelma tekee sen, mitä sen on suunniteltu tekevän, ilman virheitä.

3 Varmistus ja validointi

Kaksi samanlaista ohjelmistotestaukseen liittyvää käsitettä, joita ohjelmistotestaajat usein käyttävät, ovat todentaminen ja validointi. Molemmat käsitteet ovat luonteeltaan abstrakteja, ja kukin voidaan toteuttaa konkreettisten, suoritettavien toimintojen avulla. Nämä kaksi käsitettä selitetään seuraavasti:

3.1 Varmistus

Tällainen toiminta auttaa meitä arvioimaan ohjelmistojärjestelmää määrittämällä, täyttääkö tietyn kehitysvaiheen tuote vaatimukset, jotka on asetettu ennen kyseisen vaiheen alkua. Voidaan huomata, että tuote voi olla välituote, kuten vaatimusmäärittely, suunnitteluspesifikaatio, koodi, käyttöopas tai jopa lopputuote. Toimintoja, jotka tarkistavat kehitysvaiheen oikeellisuuden, kutsutaan verifiointitoiminnoiksi.

3.2 Validointi

Tällaiset toiminnot auttavat meitä varmistamaan, että tuote vastaa käyttötarkoitustaan. Validointitoimilla pyritään varmistamaan, että tuote vastaa asiakkaan odotuksia. Toisin sanoen validointitoiminnassa keskitytään lopputuotteeseen, jota testataan laajasti asiakkaan näkökulmasta. Validointi määrittää, vastaako tuote käyttäjien yleisiä odotuksia. Validointitoimien myöhäinen suorittaminen on usein riskialtista, koska se johtaa korkeampiin kehityskustannuksiin. Validointitoimet voidaan suorittaa ohjelmistokehityssyklin alkuvaiheessa.

4 Testauksen tyypit

Staattinen testaus: sisältää koodikatselmukset, tarkastukset ja läpikäynnit ilman koodin suorittamista.
Dynaaminen testaus: Sisältää koodin suorittamisen ja sen käyttäytymisen tarkkailun.

4.1.1 Staattinen analyysi

Kuten termi "staattinen" viittaa, se perustuu useiden asiakirjojen, nimittäin vaatimusasiakirjojen, ohjelmistomallien, suunnitteluasiakirjojen ja lähdekoodin, tutkimiseen. Perinteinen staattinen analyysi sisältää koodin tarkastelun, tarkastuksen, läpikäynnin, algoritmianalyysin ja oikeellisuuden todistamisen. Se ei sisällä kehitettävän koodin varsinaista suorittamista. Sen avulla voidaan tutkia koodia ja syitä kaikissa mahdollisissa käytöksissä, joita saattaa ilmetä ajon aikana. Kääntäjän optimoinnit ovat tavallista staattista analyysiä.

4.1.2 Dynaaminen analyysi

Ohjelmistojärjestelmän dynaaminen analyysi sisältää todellisen ohjelman suorittamisen mahdollisten ohjelmavirheiden paljastamiseksi. Myös ohjelman käyttäytymis- ja suorituskykyominaisuudet huomioidaan. Ohjelmat suoritetaan sekä tyypillisillä että huolellisesti valituilla tuloarvoilla. Usein ohjelman syöttöjoukko voi olla epäkäytännöllisen suuri.

Käytännön syistä voidaan kuitenkin valita syöttöjoukon rajallinen osajoukko. Siksi testauksessa tarkkailemme joitain edustavia ohjelmien käyttäytymistä ja teemme johtopäätöksen järjestelmän laadusta. Äärillisen testijoukon huolellinen valinta on ratkaisevan tärkeää luotettavan johtopäätöksen tekemiseksi

5 White-box ja Black-box - testaus

5.1 White-box-testaus

Määritelmä: käyttää tietoa järjestelmän sisäisestä rakenteesta sen oikeellisuuden testaamiseksi.
Tekniikat: Sisältää ohjausvirtaustestauksen, Tietovirtaustestauksen ja polkutestauksen.
Sovellus: tyypillisesti kehittäjien käyttämä oma koodinsa testaamisen.

Rakennetestauksessa tarkastellaan ensisijaisesti lähdekoodia keskittyen ohjausvirtaan ja tiedonkulkuun. Ohjausvirta tarkoittaa ohjauksen kulkua käskystä toiseen. Ohjaus siirtyy käskystä toiseen käskystä useilla tavoilla, kuten yksi käsky ilmestyy toisensa jälkeen, toimintokutsu, viestin välitys ja keskeytykset. Ehdolliset lauseet muuttavat ohjelman normaalia, peräkkäistä ohjausvirtaa. Tietovirralla tarkoitetaan arvojen etenemistä muuttujasta tai vakiosta toiseen. Muuttujien määritelmät ja käyttötavat määrittävät ohjelman tietovirran aspektin.

5.2 Black-box-testaus

Määritelmä: keskittyy ainoastaan valittujen syötteiden ja suoritusolosuhteiden tuottamiin tuloksiin, huomioimatta sisäisiä mekanismeja.
Tekniikat: Sisältää toiminnallisen testauksen, ekvivalenssiluokkien jakamisen, raja-arvoanalyysin, päätöstaulukkotestauksen ja virhearvauksen.

Toiminnallisessa testauksessa ei päästä käsiksi ohjelman sisäisiin yksityiskohtiin ja ohjelmaa käsitellään mustana laatikkona. Testauksen kohteena on ohjelmiston ulkopuolella käytettävissä olevat asiat kuten syöte ja ulkoisesti näkyvä tulos.

6 Testauksen tasot

6.1 Yksikkötestaus

Yksikkötestaus keskittyy ohjelmiston pienimpiin osiin, kuten funktioihin ja metodeihin. Testauksessa tarkistetaan, että yksittäiset koodin osat toimivat oikein itsenäisesti.

Yksikkötestaus on keskeinen osa ohjelmiston kehitystä, sillä sen avulla varmistetaan, että jokaisen komponentin toiminta on virheetöntä. Yksikkötestauksen avulla voidaan myös paikantaa nopeammin virheitä koodista. Tämän avulla saa tiedon, että jokainen pienikin osa toimii odotetusti.

6.2 Regressiotestaus

Regressio testaus on toinen testaustaso, joka suoritetaan koko ohjelmiston elinkaaren ajan. Testaus suoritetaan aina, kun ohjelmiston osia muutetaan. Sen keskeisenä ajatuksena on varmistaa, ettei mikään muutoksista ole tuonut uusia virheitä niihin osuuksiin, joita ei ole muutettu.

Tarkemmin sanottuna regressiotestaus ei ole erillinen testaustaso. Pikemminkin sitä pidetään yksikkö-, integrointi- ja järjestelmätason testauksen alivaiheena. Regressiotestauksessa uusia testejä ei suunnitella. Sen sijaan testit valitaan, priorisoidaan ja suoritetaan olemassa olevasta testitapausjoukosta, jonka avulla varmistetaan, ettei ohjelmiston uudessa versiossa ole mikään rikki.

Regressiotestaus on kallis prosessi, ja se muodostaa suurimman osan alan testausponnisteluista. On toivottavaa valita osa testitapauksista olemassa olevasta poolista kustannusten vähentämiseksi. Keskeinen kysymys on kuinka monta ja mitkä testitapaukset tulisi valita, jotta valitut testitapaukset todennäköisemmin paljastaisivat uusia vikoja.

6.3 Integraatiotestaus

Integraatiotestauksessa testataan, miten eri ohjelmistokomponentit toimivat yhdessä. Testauksen tavoitteena on varmistaa, että ohjelman osat, jotka olivat alunperin itsenäisiä, toimivat odotetusti yhdessä.

Integraatiotestausta käytetään, sillä komponenttien yhdistäminen voi tuoda esiin virheitä, joita ei ole välttämättä huomattu yksittäisten osien testaamisessa. Testauksella varmistetaan, että ohjelmisto ei vain toimi oikein osana kokonaisuutta, vaan että osat kommunikoivat keskenään oikein.

6.4 Järjestelmätestaus

Järjestelmätestauksessa testataan koko ohjelmisto kokonaisuutenaan sen toiminnallisuuksien, suorituskyvyn sekä muiden vaatimuksien osalta.

Järjestelmätestaus varmistaa, että ohjelmisto toimii kokonaisuudessaan suunnitellulla tavalla ja että kaikki osat ovat yhteensopivia. Tässä vaiheessa tarkastellaan ohjelmiston käytettävyyttä, suorituskykyä sekä turvallisuutta.

6.5 Hyväksymistestaus

Hyväksyntätestauksessa varmistetaan, täyttääkö suunniteltu ohjelmisto asiakkaan tai loppukäyttäjän vaatimukset ja odotukset. Tämä testaus suoritetaan useimmiten asiakkaan tai loppukäyttäjän toimesta.

Hyväksymistestaus varmistaa, että ohjelmisto täyttää liiketoimintatarpeet ja –vaatimukset. Se on yleisimmin viimeinen vaihe ennen ohjelmiston julkaisemista tuotantoon.

7 Testauksen automaatio

Automaatiotestaus tarkoittaa, että testaus suoritetaan ohjelmallisesti käyttämällä testausautomaatiotyökaluja ja -skriptejä. Automaatiotestit voidaan suorittaa toistuvasti ja nopeasti ilman, että testaaja itse kävisi ohjelmaa läpi.
Automatisointi on kuitenkin pitkän aikavälin investointi; se on jatkuva prosessi. Odotuksia on hallittava sen varmistamiseksi, että se on realistisesti saavutettavissa tietyn ajan kuluessa.

7.1 Testauksen automaation tärkeys

Automaatio on tärkeää, sillä se lisää testausprosessin nopeutta ja tehokkuutta. Sen avulla voidaan varmistaa testitapausten johdonmukaisen suorittamisen. Näiden lisäksi se mahdollistaa laajemman testikattavuuden.

7.2 Testauksen automaation edellytykset

Seuraavat edellytykset tulee ottaa huomioon arvioitaessa, onko organisaatio valmis testausautomaatioon:

- Järjestelmä on vakaa ja sen toiminnallisuudet ovat hyvin määriteltyjä.
- Automatisoitavat testitapaukset ovat yksiselitteisiä.
- Testaustyökalut ja infrastruktuuri ovat käytössä.
- Testausautomaation ammattilaisilla on aikaisempaa onnistunutta kokemusta automaatiosta.
- Ohjelmistotyökalujen hankintaan on varattu riittävästi budjettia.

7.3 Testauksen automaation hyödyt

Automaatiotestausta käytetään etenkin silloin, kun projekti on suuri, jossa tarvitaan monimutkaisia ja toistuvia testejä. Tämä säästää aikaa, sillä testit voidaan suorittaa nopeasti ja virheettömästi verrattuna manuaaliseen testaukseen. Automaatiotestauksella on myös se etu, että se on vähemmän altis inhimillisille virheille. Automaatio testaus mahdollistaa lisäksi tiheän sekä perusteellisen regressiotestauksen.

8 Laadunvarmistus (QA)

8.1 Laadun näkökulmat

Käyttäjän näkökulma: Laatu tarkoituksenmukaisena.
Valmistuksen näkökulma: Laatu spesifikaatioiden mukaisena.
Tuotteen näkökulma: Laatu luontaisiin ominaisuuksiin sidottuna.
Arvoperusteinen näkökulma: Laatu riippuu siitä, kuinka paljon asiakas on valmis maksamaan.

8.2 Laatutekijät ja kriteerit

Laatutekijöiden ominaisuuksia ovat oikeellisuus, luotettavuus, tehokkuus, eheys, käytettävyys, ylläpidettävyys, joustavuus, testattavuus, siirrettävyys, uudelleenkäytettävyys ja yhteentoimivuus.

Laatukriteerit ovat spesifisiä ohjelmistokehitykseen liittyviä ominaisuuksia, jotka tukevat laatutekijöitä.

8.3 ISO 9126–laatupiirteet

- **Toiminnallisuus**: Soveltuvuus, tarkkuus, yhteentoimivuus, turvallisuus, toiminnallisuuden vaatimustenmukaisuus.
- **Luotettavuus**: Kypsyys, vikasietoisuus, palautuvuus, luotettavuuden vaatimustenmukaisuus.
- **Käytettävyys**: Ymmärrettävyys, opittavuus, käytettävyys, houkuttelevuus, käytettävyyden vaatimustenmukaisuus.
- **Tehokkuus**: Aikakäyttäytyminen, resurssien käyttö, tehokkuuden vaatimustenmukaisuus.
- **Ylläpidettävyys**: Analysoitavuus, muutettavuus, vakaus, testattavuus, ylläpidettävyyden vaatimustenmukaisuus.
- **Siirrettävyys**: Mukautuvuus, asennettavuus, rinnakkaiselo, korvattavuus, siirrettävyyden vaatimustenmukaisuus.

8.4 ISO 9000:2000-ohjelmiston laatustandardi

Kyseisessä laatustandardissa keskitytään laadunhallinnan periaatteisiin. Näitä ovat muun muassa asiakaskeskeisyys, johtajuus, ihmisten osallistuminen, prosessilähestymistapa, jatkuva parantaminen sekä molempia osapuolia hyödyttävät toimittajasuhteet.

Vaatimukset määritellään laadunhallintajärjestelmälle, jossa organisaation on kyettävä osoittamaan kykynsä tarjota jatkuvasti tuotteita, jotka täyttävät asiakas- ja säädösvaatimukset.

9 POHDINTA

Ohjelmistotestaus on todella tärkeä osa ohjelmistokehitystä. Ilman sitä emme voisi varmistaa ohjelmistojemme luotettavuutta, turvallisuutta sekä toiminnallisuutta ennen niiden julkaisemista. Testauksien avulla voimme havaita mahdolliset virheet sekä puutteet, jotka muuten voisivat jäädä meiltä huomaamatta. Erilaiset testausmenetelmät, kuten staattinen ja dynaaminen, tarjoavat meille monipuolisia tapoja tarkastella ohjelmiston eri osa-alueita.

Testauksen merkitys korostuu erityisesti suuremmissa ja monimutkaisemmissa projekteissa. Näissä tapauksissa ohjelmisto sisältää useita osia, jotka on testattava erikseen ja kokonaisuutena. Integraatio- ja regressiotestaus auttavat varmistamaan, että eri komponentit toimivat yhdessä eikä ohjelmiston päivittäminen tai muuttaminen johda uusiin virheisiin.

Testauksen automatisointi parantaa testausprosessin nopeutta ja tehokkuutta. Automatisointi voi vaatia alussa investointeja ja asiantuntemusta, mutta pitkällä aikavälillä sen hyödyt ovat suuret: Testit suoritetaan nopeammin, virheettömämmin sekä suuremmalla kattavuudella. Tällä avulla testaajilla on enemmän aikaa muihin tehtäviin.

Laadunvarmistus (QA) ja ohjelmiston laatutekijöiden huomioon ottaminen ovat keskeisiä osia, jotta ohjelmisto ei ole vain teknisesti toimiva, vaan myös käyttäjälähtöinen sekä se täyttää asiakasodotukset. Tässä kohdassa ohjelmistotestauksen rooli on paljon enemmän kuin pelkkä virheiden etsiminen: se on olennainen osa kehitystä ja se tukee ohjelmiston arvoa sekä luotettavuutta.

Kaiken kaikkiaan ohjelmistotestaus on olennainen työkalu ohjelmistojen kehittämisessä. Testauksien avulla voimme parantaa ohjelmiston laatua ja suojata organisaatioita mahdollisilta riskeiltä, joita huonosti testattu ohjelmisto voisi aiheuttaa.

LÄHTEET

Naik & Tripathy: Software Testing and Quality Assurance - Theory and Practise, 2008 Wiley