Tarkvara testimise alused

Kordamisküsimused

1. Kes ja millal võttis kasutusele mõiste „*bug*“?

1870 – Thomas Alva Edison võttis kasutusele mõiste „Bug“, kui viga süsteemis

2. Milleks on vaja tarkvara testida?

Tarkvara testitakse selleks, et kontrollida, et see vastaks nõuetele. Tarkvara testimine on tarkvara kvaliteedi tagamise üks osa. Testitakse, et leida vigu.

3. Mis põhjustel võivad tekkida tarkvarra vead?

Viga on tarkvara mittevastavus nõuetele. Tarkvara analüütikud ja arendajad on ka inimesed, seega tekib vigu. Vigade tekkimine on paratamatus – keerukas süsteem, palju muudatusi, tihe projektigraafik, palju veaparandusi. Kõiki vigu pole võimalik leida.

4. Millal oleks mõistlik hakata arendatavat tarkvara testima?

Mida varem, seda parem. Mida varem vea avastada, seda odavam ja kiirem on selle parandamine ja selle uuesti testimine.

5. Millised on tarkvara testija tähtsaimad omadused?

Suhtlusoskus

6. Mis on tarkvara testimisprotsessis sisendiks, mis väljundiks?

- Sisendiks analüüs, funktsionaalsed ja mittefunktsionaalsed nõuded.
- Väljundiks vigade kirjeldused, soovitused analüüsi ja nõuete parandamiseks ning arenduse parendamiseks.

7. Kuidas toimub testimine waterfall arendusprotsessis, kuidas see toimub aga agiilses arendusprotsessis?

- Waterfall arendusprotsessis jäetakse testimine kõige lõppu. Selle tõttu leitakse vead hiljem, parandamine võtab kauem aega. See on ka kulukas. Sobib väiksematele ja mitten ii aja- ja rahakriitilistele projektidele.

- Agiilses arendusprotsessid käib töö iteratsioonides. Testimine on iga iteratsiooni osa. Testimine algab arenduse alguse ja analüüsiga. Varakult testimine aitab varem avastada kriitiisi vigu ja neid kergemini parandada.

8. Miks ja kuidas liigitatakse tarkvara testimist?

Tarkvara testimist liigitatakse omaduste järgi. Nt: kuidas süsteem peab töötama, kui kiiresti, milliseid andmeid peab tarkvara vastu võtma ja milliseid väljastama.

Kõige olulisemad:

- Funktsionaalsus
- Töökindlus
- kasutuskõlblikus
- tõhusus
- hooldatavus
- porditavus

9. Millised on tuntumad testimise liigid?

- Funktsionaalne testimine ehk vastavustestimine (functional, conformance testing)
- Jõudlustestimine (Performance testing)
- Koormustestimine (Load testing)
- Stresstestimine (Stress testing)
- Taastuvuse testimine (recoverability testing)
- Kasutatavuse testimine (Usability testing)
- Robustness testimine
- Skaleeruvuse testimine (Scalability testing)
- Turvalisuse testmine (Security testing)
- Installeeritavuse testimine (installability testing)

10. Mis on funktsionaalne testimine?

Esmärgiks on kontrollida süsteemi vastavust funktsionaalsetele nõuetele: kas vajalikke tegevusi on võimalik korrektselt läbi viia.

11. Mis on kasutatavuse testimine?

Testitakse, kas süsteem vastab temale esitatud nõuetele arusaadavuse, õpitavuse, kasutusmugavuse ning meeldivuse osas.

12. Mis on jõudlustestimine?

Annab vastuse küsimusele, milline on süsteemi toimimise kiirus tavaolukorras ning millised on kitsaskohad, mida oleks võimalik parandada.

13. Mis on turvalisuse testimine?

Kontrollida süsteemi vastavust turvalisuse nõuetele: süsteemi kasutamine peab olema võimalik ainult volitatud isikutele kindlaksmääratud osas. Kontrollida, kas on säilitatud konfidentsiaalsus, kokkukuuluvus ning kättesaadavus, kus kättesaadavus tähendab, et volitatud isikutel on infole juurdepääs, kokkukuuluvus tähendab, et info õigsus ja täielikkus on tagatud ning konfidentsiaalsus tähendab, et juurdepääs on tagatud vaid selleks volitatud isikutele.

14. Kuidas liigitatakse testimist detailsuse järgi?

- Süsteemitestimine – testitakse kogu süsteemi vastavust talle esitatud nõuetele. Musta kasti testimise alaliik. Süsteemitestimise liigid on ka vastuvõtutestimine, alfa- ja beetatestimine, kuna nende käigus testitakse kogu süsteemi.

- Integratsioonitestimine – testitakse erinevate komponentide liidestamist/koostööd. Kogu süsteemi valmisolek ei ole vajalik, sest on võimalik kasutada testidraivereid, mis simuleerivad puuduvate süsteemi osade käitumist (top-down ja bottom-up või sandwich meetodid). Suuremate süsteemide korral ongi otstarbekam liidestada süsteemid järk-järgult.

- Moodultestimine – eraldiseisva tarkvaramooduli toimimise testimiseks. Mooduli iga meetodi jaoks koostatakse test, mis on eraldiseisev mooduli teistest testidest. Moodultestimine peaks olema tarkvaraarendajate vastutusel.

15. Kuidas liigitatakse testimist läbiviija järgi?

- Alfatestimine – testimeeskonna poolt läbi viidav testimine arendaja keskkonnas arenduse lõpul

- Beetatestimine – testimine lõppkasutaja poolt väljaspool arendaja keskkonda (lõppkasutaja keskkonnas) arenduse lõpul

- Vastuvõtutestimine – testitakse süsteemi vastavust kasutaja nõudmistele (lepingu nõudmistele). Enamasti kontrollitakse vastuvõtutestimisel kasutaja tüüpilisi tegevusi. Vastuvõtutestimist võib tellija viia läbi iseseisvalt või koostöös arendajaga.

- Kasutaja-testimine – testimine lõppkasutaja poolt arenduse käigus kas arendaja või lõppkasutaja keskkonnas, kusjuures kasutaja tegevusi juhendatakse ja jälgitakse.

- In-House testimine – arendaja kasutab oma tarkvara mõnda aega enne müügile laskmist oma vajaduste rahuldamiseks, et olla kindel selle piisavas usaldusväärsuses.

- Regressioonitestimine – testimine pärast iga muudatust eesmärgiga, teha kindlaks, kas muudatused ning funktsionaalsuse lisamine pole tekitanud vigu varem toiminud programmi osas.

- Suitsutestimine – pealiskaudne testimine eesmärgiga teha kindlaks, kas pole põhjalikumat testimist takistavaid probleeme (näiteks programm ei käivitu või kõiki põhifunktsioone pole võimalik testida). Automatiseeritud suitsutestid võiksid olla esimene samm testimise automatiseerimise suunal.
- Uuriv testimine (Exploratory testing) – testimine, mille käigus järgmised testilood tulenevad eelmise tulemustest ning testija testimise ajal tekkivatest ideedest. Testilugusid pole kohustuslik dokumenteerida, kuid seda võib teha. Uuriva testimise alusena võib kasutada dokumenteeritud testilugusid, mida testija laiendab vastavalt oma mõtetele. Tegelikkuses praktiseerivad kõik testijad mingil määral uurivat testimist, kuna tavaliselt viiakse läbi rohkem testilugusid kui on dokumenteeritud. Seda põhjusel, et kõiki testilugusid ei saa paratamatult dokumenteerida. Uuriva testimise rajajaks on James Bach.

- Guerilla-testimine – uuriva testimise alaliik, mis seisneb püüdes lühikese aja jooksul leida programmi töös vigu kõige võimsamate meetoditega. Kasutatakse juhul, kui on kiiresti vaja anda hinnang programmi toimimisele. Võib eeldada, et kui kõige võimsamad rünnakud ei paljasta ühtegi viga, siis ei leidu selles kriitilisi vigu.

- Paaristestimine – paarisprogrammeerimisele sarnanev tegevus testimisel: kaks testijat kasutavad sama arvutit ning on kordamööda testimist läbiviiva ning jälgiva ja vajadusel juhendava poole rollis.

- Ad-hoc testimine – programmi mittesüstemaatiline testimine (juhuslike, mittedokumenteeritud testilugudega). Ad-hoc testimist ei ole soovitav pikka aega kasutada, kuna sellise testimise puhul pole võimalik hinnata testikatet (millised testid on tehtud), kuna mingit dokumentatsiooni läbiviidud testide kohta ei säili. Samuti ei pruugi süsteemi testikate olla piisav, kuna teste ei koostata süstemaatiliselt. Kolmandaks iseloomulikuks omaduseks on see, et kuna teste ei viida läbi süstemaatiliselt ja testidokumentatsiooni alusel, siis sõltub testimise efektiivsus oluliselt testija kompetentsusest.

16. Mis vahe on nn „valge kasti“ ja „musta kasti“ testimisel?

- Musta kasti testimise puhul koostatakse testilood süsteemile sisendite ja väljundite põhjal. Süsteemi sisemine struktuur on teadmata. Musta kasti testimismeetodite alla kuulub näiteks ekvivalentsiklasside ja piirjuhtude analüüs.

- Black Box testimine baseerub üldjuhul funktsionaalsetele nõudmistele, administratiivsetele protseduuridele ja kvaliteedi nõudmistele. Black Box testimise tehnikat kasutades vaadeldakse süsteemi kui terviklikku lahendust, millist lõpptulemusena hakatakse kasutama.

- Black Box testimise tehnikad sobivad kasutada kõikides testimise etappides alates komponentide testimisest kuni lõppkasutaja poolt tehtavale kinnitustestimisele. Kuna iga komponendi vorm on vaadeldav kui süsteemi struktuuri erinev osa, siis saame näiteks komponentide testimisel vaadelda komponenti ennast kui „musta kasti” ning testid baseeruvad komponendi funktsionaalsusele mitte aga tema struktuurile.

- Valge kasti testimise puhul koostatakse testilood programmi struktuuri põhjal. Valge kasti testimismeetodite alla käib kattekriteeriumide alusel läbi viidav testimine.

- White Box testimine baseerub lähtekoodile, programmi kirjeldusele ja tehnilisele disainile. Testimist teostatakse süsteemi sisemise struktuuri kohaste teadmiste baasil.

- White Box testimise tehnikaid saab kasutada igas testimise etapis, aga praktikas kasutatakse White Box tehnikaid ikkagi komponentide ning integratsiooni testimisel.

17. Testimise organisatoorne pool – millised on rollid ja nende ülesanded? Milliseid dokumente testimise käigus koostatakse?

Tarkvara testimise organisatoorne korraldamine on tegelikult protsess, üks tarkvara arenduse protsessi osa.

Kusjuures üks olulisemaid osasid – ilma selleta poleks tarkvara

- TESTIJUHT (test manager, test lead)
    - Aitab täpsustada nõudeid loodavale süsteemile
    - Jälgib, et nõutetest ja analüüsist peetakse kinni
    - Osaleb testiplaani koostamisel ning vajadusel algatab selle muutmist
    - Juhtib testimist kogu arenduse käigus
    - Annab õigeaegselt tagasisidet kõigist probleemidest, mis arenduse käigus võivad tekkida ning võivad nõuda testimisplaanide muutmist
    - Annab tagasisidet testimise olukorra kohta
    - Korraldab probleemide kiiremat lahendamist
    - Jälgib, et planeeritud kvaliteedi tagamise tegevused oleksid korrektselt läbi viidud

- TESTIJA
    - Testimise läbiviimine vastavalt testimisplaanile
    - Õigeaegne probleemidest teavitamine testijuhile. Probleemideks on peamiselt nõuete muudatused ning testimisel leitud kriitilised vead, mis võivad ohtu seada projekti töögraafiku täitmise
    - Tema töö on vajalik suurele osale arendusprotsessi rollidest

- Testimise käigus koostatakse:
    - Testiplaan
    - Testilugu
    - Vea aruanne
    - Testiaruanne
    - Testoaruanne

18. Mis on staatiline testimine?

- Põhiülesanne –l eida vigu juba programmi projekteerimise ja spetsifitseerimise faasides
- Saab kontrollida ka süsteemi omadusi: hooldatavus, töökindlus, analüüsitavus.
- Vähendab tarkvara arendamise kulusid ning aega.
- Ei asenda dünaamilist testimist
- Ei garanteeri, et tarkvara töötab veatult.

19. Millised on olulisemad staatilise testimise tehnikad? (Lähemalt ka sisu)

- Formaalsed
- Mitteformaalsed
- Ja nii öelda vahepealsed

20. Mis on dünaamiline testimine?

21. Olulisemad tehnikad. (Tuleb paar ülesannet)

22. Mis vahe on staatilisel ja dünaamilisel testimisel?

23. Mis on tarkvara viga?

24. Millest koosneb vea aruanne?

25. Millised on takvara arenduse ja testimisega seonduvad riskid?

26. Mida arutatakse riskianalüüsi käigus?

27. Automaattestimine – mis see on, mis on olulisem eesmärk?

28. Millistel puhkudel on automaattestimine mõistlik, millistel mitte?

29. Millistel puhkudel on manuaaltestimine mõistlikum, millistel mitte?

30. Olulisemad asjad, mid automaattestide loomise juures tähele panna ja meeles pidada?