Testiplaan

Testiplaan on projektiplaani osa ja peab olema kooskõlas projekti üldplaani ning äriliste eesmärkide ja töökorraldusega.

Testiplaan sisaldab kogu testimisprotsessi, määratledes tegevused ja andes testijaile juhised alates planeerimisest kuni testimise lõpetamiseni (planeerimine, ettevalmistamine, läbiviimine, täiustamine). Kõik testimisprotsessi tegevused võivad kattuda või toimuda samaaegselt.

Testimise korraldus peab olema läbimõeldud, planeeritud ja jälgitav.

Testiplaan on projektiülene (nt väliste arenduste korral koostab arenduspartner, täiendatakse RMIT ja MTA poolt) ning kooskõlastatud kõikide osapooltega (arenduspartner, RMIT, MTA).

Testiplaan määratleb:

- projekti üldplaanist tulenevad eeltingimused (viide dokumentidele, nõuetele, kasutusjuhtudele, skoobile, tööde üleandmise akti kinnitamise kriteeriumid)
- testimisprotsessi (meetodid, tehnikad, tegevused ja tööriistad)
    - vahendid testimiseks (nõuded tarkvarale ja seadmetele nt veebilehtede puhul, millised brauserid ja versioonid)
    - meetodid (agiilne/waterfall arendus, funktsionaalne testimine - manuaalne vs automatiseeritud testimine, MFN, turvatestimine, koormustestimine, kasutatavuse testimine jne)
    - testilugude formaat
- testimise ulatuse ja eesmärgid (lähtuvalt prioriteetidest, mida ja kui palju testidega kaetakse. PS! kindlasti tuleb läbida protsessid, mis on osa tavapärasest tööprotsessist)
    - mida testitakse (kombinatsioonid ideede tasemel) ja mida ei testita (koos põhjendustega)
    - eesmärgid, mida on vaja saavutada
- testimistegevused, nende tegevuste oodatud tulemid ja ajalised hinnangud (etapiliste arenduste korral etapiti välja tooduna)
    - ootused ja kvaliteedinõuded on täpsustatud, dokumentatsioon testitakse testiplaani koostamise käigus, kui pannakse paika testiideed /testilood/regressiooni lood – vajadusel koostöös analüütiku/arendusmeeskonnaga
    - erinevate ülesannete eeldatavad testimise ajamahud (sh. testjuhtumite kirjeldamise ja täiendamise aeg, regressioonitestide maht)
- ajakava (kooskõlas projektiplaaniga, ressursside ja tarnest lähtuva ajakavaga)
        - testimisele planeeritud aeg ressursside lõikes
        - suitsutestide, koormustestide, regressioonitestide, keskkonna katkestuste jm planeeritud ajad
        - varuaeg, lähtudes muudest ajalistest piirangutest (paigaldus viibib, puhkused, pühad jne)
        - planeeritud aeg etapiti vigade paranduseks arenduspartnerile
        - aeg vigade paranduste kontrolliks ning regressiooniks
        - suurte projektide korral vahearuannete esitamise ajad (kvaliteedi ja ajakavas püsimise hetkeseisu fikseerimise eesmärgil)
- testijate ressursside jaotus, koos vastutustega (kõik osapooled peavad teadma oma kohustusi ja vastutust, koolitus- ja juhendamisvajadused)
- testimiseks vajalikud kasutajaõigused ja privileegid (alati projektipõhiselt esitatud nii uute arenduste kui ka muudatuste korral)
- eeldused ja piirangud – nõuded testandmetele (millised andmekomplektid, sisendid jne, lähtudes SKO poolsetest nõuetest) ja testikeskkonnale (testkeskkond peab olema võimalikult sarnane lõppkeskkonnale, et vältida keskkondade erinevusest tulenevaid probleeme)
- kui arendatakse infosüsteeme, mis kasutavad teiste infosüsteemide teenuseid, siis funktsionaalne testimine peab hõlmama ka liidestatavate infosüsteemide seotud funktsionaalsuste testimist (katkestused liidestustes kooskõlastatud)
- riskid - oletatavad riskid, mis võivad tekkida ja millise tõenäosusega – mõju ning võimalikud tegevused nende realiseerumisel
- tootestamise ja vastuvõtmise otsustamise kriteeriumid (kriteeriumid, mille puhul võib lugeda arenduse edukaks ja jätkusuutlikuks või mitte)
    - kasutatavus – tõhusus, hõlpsus
    - arusaadavus – õpitavus, mõistetavus
    - funktsionaalne sobivus (ootustele vastavus) – toimib äriloogiliselt õigesti ja ei sisalda funktsionaalsust, mida pole tellitud/soovitud
    - vastuvõtu akti kriteeriumid - milliste puudustega loetakse töö vastuvõetuks, milliste puhul mitte
- infovahetuse kontaktid (arendaja, äri ja RMIT, tihedus ning kanalid, sh. testandmete ettevalmistus, vigade parandus, testpostkast veateadete jaoks)
- testimise väljundid (testi vahearuanded, testiraport)
- testimise lõpetamine – testimistegevuste/õppetundide analüüs, hinnangud ja järeldused

Testiplaani tuleb vajadusel jooksvalt muuta (probleemide tuvastamisel, nõuete muutumisel, ajagraafikust mitte kinnipidamisel), vastav info tuleb koheselt esitada teistele osapooltele.

Testiplaan peab olema kirjalikku taasesitamist võimaldaval kujul Confluence's (igal testiplaanil on unikaalne identifikaator ja on see seotud testilugude ning alamplaanidega nende olemasolul nt regressiooniplaan, koormustestiplaan jne).

Testilugu

Testilugude koostamine

Testilugu on detailne kirjeldus, kuidas ja läbi milliste tegevuste toimub mingi süsteemi omaduse (vastavus vajadustele ja kokkulepitud funktsionaalsus, tegevusi on võimalik korrektselt ja mõistlikult läbi viia) toimimise kontrollimine ning otsustamine, kas süsteem toimib korrektselt või mitte. Testilood koostatakse süsteemi testitavate nõuete põhjal (funktsionaalsed ja mittefunktsionaalsed nõuded, spetsifikatsioonid, testiplaan, turvanõuded, kasutusjuhud). Kuna kõiki testilugusid ei ole võimalik kohe ette näha ja kirjeldada, siis peaks need olema loogiliselt struktureeritud, tagamaks parema jälgitavuse (kui palju kaetud/läbitud/ei vasta nõudmistele) ja hallatavuse (nõude muutumisel täiendada/parandada jooksvalt).

Testilugude koostamise asemel võib ka kasutada testiideede kirjeldamist (põhipunktid, mida peaks testima või millele tähelepanu pöörama).

Testilugude koostamine, täiendamine ja hoiustamine toimub kirjalikult taasesitamist võimaldaval kujul Confluence's. Testilood peavad olema kirjeldatud arusaadavalt, et need oleks kasutatavad vajadusel ka regressioonitestimiseks.

Testilugude koostaja:

- Arenduspartneri poolt teostatud arendusega esitab partner testilugude kirjeldused (nii kasutajaliidese (UI) kui X-tee lahendus (S2S). Vastavalt vajadusele kirjeldatakse puuduolevad juurde)
- Sisearenduste korral tuleb testija/testijuhi poolt testilood kirjeldada
- Riikidevahelise infovahetuse arenduste puhul on Euroopa Komisjoni poolt on ette kirjeldatud testjuhtumid (vastavalt vajadusele tuleb testija /testijuhi/IT arenduspartneri poolt siseriiklikud juurde kirjeldada)

Testilugude koostamiseks:

- Testilugusid tuleb kirjutada põhimõttel, et nii palju kui vajalik ning nii vähe kui võimalik, lähtuvalt funktsionaalsuse muudatusest ja riskidest
- Testilugudega on kaetud kõik kasutuslood, mida tehakse igapäevaselt (erineva nurga alt, teekondi pidi, tegevuste järjestusi pidi, eri õigustes kasutajatega)
- Testilugudena peavad olema kaetud nii positiivsed kui ka negatiivsed stsenaariumid
- Sama testiloo kordamisel teha seda uue sisendiga (leidmaks potentsiaalseid vigu andmeerisusest)
- Testilugusid tuleb pidevalt täiendada ja muuta kuni projekti lõpuni
- Tehniliste testide (sh. liideste) testimisel tuleb vajadusel kaasata RMIT rakenduse administraator
- Testija peab aru saama äriprotsessidest (tundma sisu) või testilood peavad olema vastava oskusega inimese poolt valideeritud

Testilugu peab sisaldama järgnevat:

- unikaalne ID – testiloo identifitseerimiseks
- seotud süsteemid (nt äriregister)
- eeltingimused (nõuded testandmetele ja eeltegevustele)
- erinõudmised keskkonnale või kasutajarollile (kasutaja näeb ja saab muuta vaid rollile ettenähtut)
- viide analüüsile
- lühikirjeldus – testitav omadus lühidalt, et testija saaks kiiresti ettekujutuse, millise omaduse testimist kirjeldatakse
- tegevused samm-sammult (sisendid)
- tegevustele vastavad oodatavad tulemused
- testimise tulem info, kas on korras või mitte

Vigade raporteerimine

Tarkvara mittevastavus nõuetele ja/või vajadustele ehk viga tuleb raporteerida võimalikult täpselt kirjeldades vea olemus ja selle tekkimise protsess, et selle järgi oleks võimalik viga taas tekitada, tuvastada vea põhjus ja seejärel parandada.

Tuvastatud vead (iga viga eraldi) raporteeritakse alati vastavalt kokkulepitule JIRA-sse (JIRA-s välja „Projekt“ väärtus on nt 'e-MTA töölauad (MA0112)' ja „Tüüp“ väärtus on 'Viga'), kus viga sisaldab alati järgnevat infot:

- unikaalne ID (automaatne numereerimine JIRA-s nt …-… Vea number)
- vea prioriteet (JIRA-s väli "Prioriteet“, väärtus valida vastavalt vea mõjule)
    - **Blokeeriv/takistav viga (Blocker)** – funktsionaalsuse kasutamine on täielikult takistatud, puudub WAR (vajab kohest parandamist, et töö saaks jätkuda. Arendaja reageerib koheselt, katkestades muud tegevused, hotfix esimesel võimalusel)
    - **Kriitiline viga (Critical)** – funktsionaalsuse kasutamine on märkimisväärselt takistatud (vajab kiiret parandust, töö on olulisel määral takistatud. Arendaja reageerib koheselt, parandus esimesel võimalusel või järgmise tarnega sõltuvalt kokkuleppest äri/RMITiga)
    - **Oluline viga (Major)** – funktsionaalsuse kasutamine on häiritud (vajab kiiret parandust, kuid selle tõttu ei ole töö takistatud. Arendaja parandab vea järgmise korralise tarnega)
    - **Väike viga (Minor)** – viga ei takista funktsionaalsuse kasutamist nt visuaali vead (ei vaja kiiret sekkumist, aga vajab parandamist. Arendaja parandab vea järgmise korralise tarnega)
    - **Triviaalne/väheoluline (Trivial)** – viga ei takista funktsionaalsuse kasutamist nt tähevead (parandatakse kui kõik muu on korras või vastavalt kokkuleppele arendaja ja RMIT/MTA vahel)
- "Keskkond", mis keskkonnas probleem esineb (Tootekeskkond/Testkeskond/Pre-live keskkond, kui mitmes keskkonnas, siis loeteluna)
- "Omanik" - suunamine algselt kokkulepitud isikule (üldjuhul testijuhile), kes valideerib vea (kas tegu vea või uue tööülesandega) ja veakirjelduse ning suunab edasi arendaja kontaktisikule projektis või rakenduse administraatorile, taski täiendamiseks (logid, baasi väljavõtted).

Veakirjeldus (JIRA-s väli „Kirjeldus/Description“) peab sisaldama:

- viide testiloole/tegevusele, mille läbimisel viga tekkis (või analüüsipunktile, millega vastuolu) – testijuhtumi tunnus nt UC1 TC01-01
- vea pealkiri (JIRA-s väli „Kokkuvõte“) võimalikult selge ja ülevaatlik (max 80 tähemärki ehk üks rida)
- probleemi detailne kirjeldus (konkreetne, arusaadav ja neutraalne) (sh. vea kirjeldus)
- samm-sammult tegevused ning detailsed tegelikud tulemused, koos sisend- ja väljundandmetega (nt. avasin '…' > vajutasin '…' > sisestasin '…' > …)

Vea taastekitamiseks/kirjeldamiseks on vajalik anda sisendiks maksimaalselt infot (kuidas taastekitada/põhjuse lokaliseerimine - viga on konfis /koodis/teenustes/disainis/algandmetes):

- alati lisada võimalikult täpne URL (peab olema lihtsalt arusaadav, kus keskkonnas ja millises alamjaotises viga esines)
- mis seadme ja brauseriga viga esineb (eelkõige disaini probleemide korral)
- autentimine (sisse loginud kasutaja isikukood, roll ja edasine detailsem õiguste valik kui see on tehtud)
- kasutatud andmed nt lepingu number, numbrilised väärtused vms (võimalikult täpne andmekomplekt, et selle järgi on võimalik leida logid jm)
- täpsed järgnevad sammud, mida tuleb teha vea taastekitamiseks - mida ja mis järjekorras klikkida
    - vea kirjeldus ja ekraanipilt/video
    - vea püsivus (kas viga tekib alati ja kohe või …?)
    - vea korratavuse hinnang (ka vead, mida ei õnnestu korrata või mis näiliselt ilmnevad juhuslikult, tuleks dokumenteerida koos kordamise katsetustega (mida on tehtud, et proovida viga esile kutsuda))
    - vea mõju (kui paljusid kasutajaid ja millist osa funktsionaalsusest hõlmab, millised on järeltegevused, kui viga ei saa kohest parandust)
- täpne veatekkimise aeg (logikirjete ja andmebaasi väljavõteteks ning kontrolliks)
- oodatud tulemuse kirjeldus ehk kirjeldus selle kohta, mis pidi juhtuma, kuid ei juhtunud
- WAR (Work Around Solution ehk alternatiivne lahendus) - olemasolul, võimalikult täpselt kirjeldatuna

Viga tuleb linkida vajadusel peataski külge.

JIRA-sse lisatud vea link lisatakse alati confluence tabelisse ja vastavalt kokkuleppele ka vajadusel Skype vestlusesse.

Veaparanduse testimine

Arenduspartner suunab veaparanduse (paigaldamiseks ning) testimiseks, lisades veataskile kirjelduse:

- mis oli vea põhjus
- kas parandati ainult konkreetset juhtu või vaadati üle kogu sarnane funktsionaalsus
- millised testid on peale parandust teostatud

Kirjelduse detailsus sõltub vea keerukusest (lihtsad vead nt tähevead jne ei vaja detailset arendaja poolset kirjeldust).

Veaparanduse üle testimise järel tuleb JIRA-s sõltuvalt testitulemusest:

- viga ei ole parandatud
    - lisada vastavasisuline kommentaar uue näite/infoga
    - märkida staatuseks ’Vaja teha’
- viga on parandatud
    - lisada ’Kommentaar’ väljale selge kontrollitav viide projektile/testiraportile ja tulemus, kuidas lahendus toimib
    - märkida staatuseks ’Suletud’
- viga on parandatud, aga üle testimise käigus tuvastati uus viga
    - tuvastatud mittevastavus kajastada uue veataskina JIRA-s
    - lisada testitud taski ’Kommentaar’ väljale selge kontrollitav viide projektile/testiraportile ning tuvastatud veale
    - olemasolev veatask märkida staatusesse ’Closed’ ehk ’Suletud’

Testiraport (lõppraport)

Peale testimistegevuste lõpetamist (arendatud, testitud, vead parandatud ja üle testitud, regressioonitestimine teostatud) koostatakse teostatud testimise kohta kokkuvõtlik testiraport, mis lisatakse (viitega raporti asukohale Confluence's) JIRA taskile, mille raames testimine teostati. Väiketöö või vigade paranduste testimise korral piisab JIRA taski alla lisatud kommentaarist (sisaldab testinäiteid, ülevaadet testimise seisust), mis antud juhul asendab testiraportit.

Testiraport peab sisaldama järgnevat infot:

- keskkond, kus testiti (riist-ja/või tarkvara - teadaolev tehniline eriseadistus, testseadmed ja brauserid, mida testimisel kasutati ning testitavate keskkondade URLid)
- milliste kasutajate ja/või rollidega testiti (kliendi ja/või ametniku vaade, klienditüübi kombinatsioonid)
- põhinõuded (viide dokumentatsioonile, peamised testiideed/kasutuslood)
- kokkuvõte, mida ja kuidas testiti (milliseid testimistehnikaid/meetodeid kasutati) - testiplaani järgimine
- kokkuvõte testitulemustest (nõuetele vastavus ja kõrvalekalded ehk oodatud tulemus vs tegelik ning hinnang kasutatavusele)
- tuvastatud vead ja probleemid:
    - raporteeritud vead (koos viitega JIRA taskile ja kommentaaridega taski staatuse kohta)
    - lahendamata vead, feature-d ehk süsteemi käitumise iseärasused ja probleemid (kommentaaridega, kuidas vältida ja WAR (workaround) ehk ajutise lahenduse infoga, selle olemasolul)
    - märkused – millele tuleks tähelepanu kindlasti pöörata
- hinnangud
    - mida ja kui põhjalikult testiti ning mida ei testitud ja miks
    - projekti kvaliteedile ja probleemkohtadele
- ülevaade riskidest (oletatavad riskid, mis võivad tekkida ja millise tõenäosusega – mõju ning võimalikud tegevused nende realiseerumisel)

Testiraporti põhjal peab olema tuvastatav: millal testiti, kelle poolt ja milliste andmetega, et vajadusel kontrollida, kas on testitud ja kuidas funktsionaalsus sel hetkel toimis.

Testiraport peab olema loetav ja arusaadav ka teistele osapooltele.