Ville Kari

Robot Frameworkin validointi

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
18.5.2011

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Ville Kari Robot Frameworkin validointi 26 sivua 18.5.2011
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	pääsuunnittelija Tero Kinnunen yliopettaja Markku Karhu

Insinöörityön tavoitteena oli validoida Robot Framework -työkalu GE Healthcare Finlandille. Validointi on prosessi, jolla varmistetaan, että työkalu täyttää yrityksen laatujärjestelmän sille asettamat kriteerit. Robot Framework on testausautomaatiotyökalu, jolla voidaan suorittaa hyväksyntätason testausta. GE Healthcare Finlandilla työkalua tullaan käyttämään sairaaloissa käytettävän potilasvalvontajärjestelmän testauksen osittaiseen automatisointiin.

Insinöörityön aikana tutustuttiin yrityksen laatujärjestelmän määrittelemään validointiprosessiin ja sen pohjalta määriteltiin vaadittavat työvaiheet ja dokumentaatio. Ensimmäiseksi työssä luotiin validaatiosuunnitelma, jota seuraamalla validaatio suoritettiin. Työkalulle luotiin käyttäjä- ja järjestelmävaatimukset, joilla varmistettiin, että työkalu tukee yrityksen näkökulmasta haluttuja asioita, kuten testiajojen ilmoittaminen joko onnistuneeksi tai epäonnistuneeksi, tuki käyttäjän omille kirjastoille sekä yhteenvetoraportin automaattinen luominen. Vaatimusten pohjalta luotiin testejä, joilla kaikki vaatimukset saatiin katettua. Lisäksi luotiin ylimääräisiä testejä, joilla varmistettiin, että työkalu toimii kuten lupaa ja sen tuottamat testiraportit ovat luotettavia.

Validointiprosessin aikana ei työkalusta löydetty yhtään poikkeamaa, joka olisi vaatinut erikoisempia toimenpiteitä tai työkalun käyttöönoton hylkäämistä.

Insinöörityön lopputuloksena syntyi tarvittava validointidokumentaatio ja laatuvastaavalta taholta saatiin hyväksyntä työkalun käytölle yrityksessä.

Avainsanat	Robot Framework, validointi, automaatiotestaus

Metropolia

Abstract

Author(s) Title Number of Pages Date	Ville Kari Validation of Robot Framework 26 pages 18 May 2011
Degree	Engineer of Applied Science
Degree Programme	Information Technology
Specialisation option	
Instructor(s)	Tero Kinnunen, Lead System Designer Markku Karhu, Principal Lecturer

The goal in this thesis project was to validate Robot Framework test automation tool for company named GE Healthcare Finland. Validation is a process to ensure that the tool fulfills company's quality requirements. Robot Framework is a test automation framework for running acceptance level tests. In GE Healthcare Finland tool is going to be used in automating parts of the patient monitoring system testing.

Needed validation effort and documentation was defined by getting familiar company's quality policy and design control methods. Validation plan document was created and that worked as a guide for the rest of the process. User and system requirements were defined for the tool to be sure that tool works as wanted. Based on the requirements test cases were written and executed to be sure that all requirements were fulfilled. Addition to all validation effort some additional tests were created to confirm that the tool works as promised and results that it creates are reliable.

During the validation process no defects were found that would have required further inspection or activities or even rejecting the use of tool.

As a result of thesis project the needed validation documentation were created and quality representative approved the use of the tool in GE Healthcare.

Keywords	Robot Framework, validation, test automation

Sisällys

1 Johdanto

Tämän insinöörityön aiheena on Robot Framework -työkalun validointi GE Healthcare Finlandille. Validointi on prosessi, jolla varmistetaan, että työkalu täyttää sille asetetut kriteerit. GE:n laatupolitiikka vaatii, että kaikki tuotekehityksessä ja testauksessa käytettävät työkalut on validoitu, jotta voidaan olla varmoja, että ne täyttävät ennalta määritetyt tehtävänsä. Koska GE Healthcare Finland toimii lääketieteen alalla, tehdään ulkopuolisen toimitsijan toimesta säännöllisin väliajoin auditointeja joissa tarkistetaan että laatujärjestelmät ovat kunnossa.

GE Healthcare Finland on lääketieteen alan tuotteita valmistava yritys. Aikaisemmin Helsingin yksikkö tunnettiin nimellä Instrumentarium Datex-Ohmeda, kunnes monikansallinen yritys General Electric osti sen vuonna 2003. GE Healthcare valmistaa potilasvalvontamonitoreita, kliinisiä tietojärjestelmiä sekä kuvantamislaitteita, kuten röntgen-, magneettikuvaus-, tietokonetomografia- sekä ultraäänilaitteita. [1.]

Robot Frameworkin validoinnin tilasi Mobile Viewers projekti, joka tulee ensimmäisenä yrityksessä käyttämään työkalua. Mobile Viewers tuotteet laajentavat sairaalan potilasvalvontajärjestelmän eri päätelaitteille sekä tarjoavat hakulaiterajapinnan. Mobile Viewers tuotteen keskeinen komponentti on Mobile Care Server, keskuspalvelin, joka pyörii Red Hat linux -käyttöjärjestelmällä varustetulla PC:llä. Mobile Care Server vastaanottaa potilastietoja monitorointiverkoista ja välittää niitä eteenpäin päätelaitteille. Mobile Viewers tarjoaa Java-pohjaiset sovellukset tietokoneille, matkapuhelimille sekä kämmentietokoneille.

Insinöörityön tavoitteena on saada Robot Framework -työkalu validoitua, jotta Mobile Viewers 6.0 kehitysprojekti voi osittain automatisoida hakulaiterajapintansa hyväksyntätestauksen.

2 Robot Framework

Robot Framework on avainsanoilla ohjattu testiautomaatioympäristö hyväksyntätason testaukseen ja hyväksyntätestausohjattuun kehittämistyöhön. Robot Framework on open source –ohjelma, jota julkaistaan Apache License 2.0 -lisenssin alaisena, ja sen tekijänoikeudet omistaa Nokia Siemens Networks.

Robot Framework on ohjelmoitu käyttäen Python-ohjelmointikieltä, ja sen ominaisuuksia voidaan laajentaa käyttäjän omilla Python- ja Java -testikirjastoilla. Käytettäessä Java-kirjastoja Robot Frameworkia ajetaan Jython-tulkkia hyväksikäyttäen. [2.]

Testit määritellään taulukkomaisesti jäsenneltyinä joko HTML-, TSV- (Tab Separated Values), reST- (reStructuredText) tai tekstitiedosto (.txt) -formaatissa. HTML-formaatissa testidata määritellään omiin taulukkoihinsa, asetuksille, muuttujille, testitapauksille ja käyttäjän omille avainsanoille kullekin omansa (taulukko 1).

Taulukko 1. Esimerkki HTML-formaatista.

Muuttuja	Arvo	Arvo
${var}	Terve, maailma!	

Testitapaus	Toiminto	Argumentti
Minun testi	[Dokumentaatio]	Esimerkki
	Avainsana 1	${var}
Toinen Testi	Avainsana 2	${var}

TSV-formaatissa kaikki testidata on yhdessä taulukossa eroteltuna toisistaan sarkaimin. Otsikot ja testitapaukset on eroteltu toisistaan asteriksien (*) avulla. Visuaalisesti TSV ei ole kovin hyvä valinta, koska yhden sarkaimen käyttö yksiköiden välissä tekee lopputuloksesta usein kovin sekavan (taulukko 2).

Taulukko 2. Esimerkki TSV-formaatista.

```
*Muuttuja* *Arvo*     *Arvo*
${var}     Terve, maailma!

*Testitapaus*         *Toiminto* *Argumentti*
Minun testi           [Dokumentaatio]       Esimerkki
          Avainsana 1           ${var}

Toinen testi          Avainsana 2           ${var}
```

reST-formaatissa testidata on jäsennelty helppolukuiseksi yhtäsuuruusmerkkien ja välilyöntien avulla (taulukko 3).

Taulukko 3. Esimerkki reST-formaatista.

```
================  ====================  ==================
   Muuttuja             Arvo                  Arvo
================  ====================  ==================
${var}            Terve, maailma!
================  ====================  ==================


================  ====================  ==================
  Testitapaus           Toiminto            Argumentti
================  ====================  ==================
Minun testi       [Dokumentaatio]       Esimerkki
\                 Avainsana 1           ${var}
\
Toinen Testi      Avainsana 2           ${var}
================  ====================  ==================
```

Edellä mainittujen formaattien lisäksi on myös niin kutsuttu puhdas teksitiedostoformaatti, jossa testidata on jäsennelty joko välilyöntien tai putkimerkkien avulla. Putkimerkein eroteltu testidata on huomattavasti helppolukuisempi (taulukko 4).

Taulukko 4. Esimerkki puhtaasta tekstitiedostoformaatista.

```
| *Muuttuja* | *Arvo*
| ${var}     | Terve, maailma!

| *Testitapaus* | *Toiminto*      | *Argumentti*
| Minun testi   | [Documentaatio] | Esimerkki
|               | Avainsana 1     | ${var}
| Toinen Testi  | Avainsana 2     | ${var}
```

Robot Framework -testit perustuvat avainsanoihin, joilla ohjataan testejä sekä tehdään muuttujien vertailuja ja muokkauksia. Avainsanat palauttavat aina tiedon siitä,

onnistuiko avainsanan suoritus tai mikä oli vertailun tulos, tosi vai epätosi. Robot Frameworkissa on paljon sisäänrakennettuja avainsanoja, joilla onnistuvat erilaiset muuttujien ja datan vertailu ja muokkaaminen sekä testiajon ohjaus. Käyttäjät voivat muodostaa omia korkeamman tason avainsanoja keräämällä yhteen matalamman tason avainsanoja, näin testitapaukset säilyvät helppolukuisina, kun toistuvat rutiinit ja monimutkaiset testit voidaan piilottaa omiin avainsanoihinsa.

Automaatiotestaus Robot Frameworkilla

Robot Framework tarjoaa mahdollisuuden hyvin selkeiden testien kirjoittamiseen avainsanapohjaisella lähestymisellään. Itse testitapaukset voivat olla hyvin korkean tason tyyppisiä, ja monimutkaisemmat toiminnallisuudet, skriptit ja ohjelmakoodit voidaan piilottaa alemmille tasoille avainsanojen taakse tai testikirjastoihin. Jos testattavassa tuotteessa muuttuu jokin tekninen toteutus, ei korkean tason testejä tarvitse muuttaa vaan riittää, kun muutos tehdään matalammalle tasolle. On myös mahdollista, että testisuunnittelijat kirjoittavat korkeantason testit valmiiksi vaatimusten pohjalta jo ennen kuin tekninen totetutus on valmis tai se on jopa suunnittelematta.

Robot Framework tukee sekä käytösohjattuja, dataohjattuja että avainsanaohjattuja testitapauksia. Käytösohjatut testit ovat hyvin helppolukuisia ja testit ovat käytännössä vaatimuksia, joissa määritellään alkutilanne, tapahtuma ja lopputulos. Taulukossa 5 on esimerkki käytösohjatusta testitapauksesta, jossa testataan, että käyttäjä pääsee hallintasivuille syöttämällä oikean salasanan.

Taulukko 5. Esimerkki käytösohjatusta testistä.

Test Case	Steps
User can login to administration-pages	Given user has a valid account
	when he inputs correct password to login page
	then administration-pages are displayed

Käytösohjatussa testauksessa testitapaukset luodaan usein ennen kuin testattavaa ominaisuutta on edes luotu tuotteeseen. Käytösohjatut testit ovat oivia eri sidosryhmien väliseen kommunikaatioon, koska ne tukevat vapaata tekstin käyttöä ja

ovat siksi helppolukuisia. Ajettaessa testiä etuliitteet *given*, *when*, *then* ja *and* jätetään huomioimatta ja loppuosa muodostaa käytetyn avainsanan.

Dataohjatuissa testeissä ajetaan usein yksi testitapaus useaan kertaan eri syötteellä. Dataohjatut testit eivät välttämättä ole aivan yhtä helppolukuisia kuin käytösohjatut testit. Taulukossa 6 on esimerkki dataohjatusta testitapauksesta, jossa testataan käyttäjän antamaa salasanaa hallintasivuille ja riippuen salasanan oikeellisuudesta ladataan joko hallintasivut tai virhesivu.

Taulukko 6. Esimerkki dataohjatuista testeistä.

Test Case	Keyword	Password	Loaded page
Correct password	Login with password	AbCd123	Administration page
Empty password	Login with password		Error page
Wrong password 1	Login with password	abcd123	Error page
Wrong password 2	Login with password	ABCD123	Error page
Wrong password 3	Login with password	qwerty	Error page

Dataohjatuissa esimerkkitesteissä on vain yksi avainsana, "Login with password", jota käytetään useammassa testitapauksessa eri syötteillä. Ylläpidon kannalta sisäänkirjautumislogiikan piilottaminen yhden avainsanan taakse tekee testien päivittämisen helpoksi; jos jokin logiikassa muuttuu, tarvitsee vain yhtä kohtaa testeistä päivittää.

Avainsanaohjatut testit ovat hyvin samanlaisia kuin käytösohjatut testit. Yleensä testit sisältävät alkutilanteen määrittelyn, tapahtuman ja lopputuloksen. Avainsanaohjatuissa testeissä on tärkeää käyttää mahdollisimman kuvaavia avainsanoja sekä pitää korkein taso selkeänä ja helppolukuisena. Taulukossa 7 on esimerkki avainsanaohjatusta testistä, jossa alkutilanteeksi avataan kirjautumissivu, syötetään validi käyttäjätunnus sekä salasana ja kirjaudutaan sisään. Lopuksi tarkistetaan, että sisäänkirjautuminen onnistui.

Taulukko 7. Esimerkki avainsanaohjatuista testeistä.

Test Case	Keyword	Argument
Valid Login	Open Login Page	
	Input Name	user
	Input Password	secret
	Submit Name and Password	
	Welcome Page Should Be Open	

3 Automaatiotestauksen haasteet

GE Healthcarella järjestelmätason testaus tehdään käytännössä pelkästään manuaalisesti testaajien toimesta. Jos vaatimus on sen luontoinen, että sitä on mahdotonta verifioida testaamalla, verifioidaan vaatimus koodikatselmuksella, jossa ohjelmoijat analysoivat ohjelmakoodia ja varmistavat, että se on toteutettu oikein.

Manuaalisella testauksella on sekä hyvät että huonot puolensa. Hyvää on esimerkiksi se, että voidaan havaita poikkemia varsinaisen testauksen alla olevan kohteen ulkopuolelta, kun taas automaattiset testit tarkistavat vain ennalta määritetyt asiat. Huonoja puolia manuaalisessa testauksessa ovat sen hitaus ja kaikkien mahdollisten kombinaatioiden testaus, kun taas automaattinen testaustyökalu tekee työtä ilman taukoja yötä päivää.

Automaatiotestauksen suurimpia haasteita on testien ylläpidettävyys. On ensisijaisen tärkeää, että testit ovat helppolukuisia ja itsensäselittäviä. Jos testit ovat vain skriptejä tai ohjelmakoodia, ei niiden ylläpito onnistu kuin niiden kehttäjiltä. Automaatiotestit tulisi kirjoittaa niin, että kuka tahansa voi saada selvää niiden tarkoituksesta ja toiminnasta kuukausien tai vuosien jälkeenkin.

4 Työkalun validointiprosessi

4.1 Validointiprosessin periaatteet

Työkalun validoinnissa on seurattava yrityksen laatimaa ohjeistusta laatukäsikirjasta. GE:llä on tarkka validointiprosessi tietokoneavusteisille järjestelmille, joita käytetään tuotekehityksessä ja testauksessa.

Jotta terveydenhuoltoalan tuotteita voidaan ylipäätänsä myydä maailmanlaajuisilla markkinoilla, tulee yrityksen laatujärjestelmien olla FDA:n (US Food and Drug Administration) hyväksymät. FDA auditoi säännöllisin väliajoin, että GE täyttää sen asettamat laatustandardit.

FDA on Yhdysvaltain elintarvike- ja lääkevirasto, joka laatii säädökset Yhdysvaltojen markkinoilla myytäville tuotteille. Jokaisen lääketieteellisen laitteen on noudatettava FDA:n säädöksiä, jos sitä halutaan myydä Yhdysvaltojen markkinoilla. FDA:n säädökset eivät rajoitu pelkästään yrityksen laatujärjestelmään vaan niihin sisältyvät sen lisäksi muun muassa:

- tuotanto- ja jakeluyksiköiden rekisteröinti ja listaus
- lääketieteellisten laitteiden listaus
- esimarkkinointi-ilmoitus
- lupa käyttää laitetta kliinisiin tutkimuksiin
- tuotteiden merkintävaatimukset. [3.]

Koska FDA:n säädösten tulee päteä hyvin monille toisistaan erilaisille tuotteille ja yrityksille, FDA ei määrittele säädöksiään kovin tarkasti. FDA määrittelee ennemminkin yritysten omille laatujärjestelmille kehykset, joita yritykset voivat tarkentaa sen mukaan, minkä alan tuotteita ne valmistavat. FDA:n säädösten ja vaatimusten tarkoitus on tuottaa markkinoille turvallisia ja tehokkaita tuotteita. [4.]

4.2 GxP-dokumentti

Ensimmäinen askel validoinnissa on GxP-dokumentin täyttäminen ja sen lopputuleman perusteella, hyväksyttäminen laatuvastaavalla. GxP on lyhenne sanoista Good x Practicies, jossa x-kirjain edustaa sanoja kliininen, jakelu, laboratorio sekä tuotanto.

GxP-dokumentin avulla määritellään työkalun käyttötarkoitus sekä se, täyttääkö työkalu ennalta määritellyt ehdot, jolloin työkalu on validoitava. Ennalta määritellyt ehdot koskevat sähköisten tallenteiden luomista, käsittelyä ja säilytystä sekä laatuprosessien automatisointia.

Koska Robot Framework tekee itsenäisesti verifikaation hyväksymis- tai hylkäyspäätöksen ja luo testin päätteeksi elektronisen tulosraportin, joka tallennetaan suoraan laatujärjestelmään, GxP-tulokseksi tuli positiivinen; validointi vaaditaan.

Jos GxP-tulos olisi ollut negatiivinen, olisi tämä tulos pitänyt hyväksyttää laatuvastaavalla ja työkalulle olisi riittänyt paljon muodollisempi validointi. Laatuprosessit eivät vaadi minkäänlaista työkalun validointia jos GxP-tulos on negatiivinen, mutta GE Healthcarella on ollut tapana validoida kaikki käytössä olevat työkalut, tosin hieman vapaamuotoisemmin.

4.3 Ohjeistuksen valinta

Koska työkalujen validointiprosessin ohjeistus on vasta muutaman vuoden vanha, on vanhoille jo käytössä oleville työkaluille oma ohjeistuksensa sekä uusille ensimmäistä kertaa käyttöönotettaville omansa. Robot Framework on uusi työkalu, joten validointiprosessissa tulee seurata uusille työkaluille tarkoitettua ohjeistusta.

4.4 Validointisuunnitelma

Kun GxP-tulos on selvillä on seuraavana työlistalla Validation Plan -dokumentaatio jossa määritellään kaikki työkalulle tehtävät validointi toimenpiteet. Työkalun validointiohjeistus määrittelee tarkasti mitä kaikkea Validation Plan -dokumentaatiosta

on löydyttävä. Seuraavissa kappaleissa jokainen vaadittava kohta esitellään tarkemmin ja kerrotaan, mitä Robot Frameworkin tapauksessa dokumenttiin kirjattiin.

Roolit

Ensimmäinen tehtävä on määritellä validoinnissa mukana olevat ihmiset tai ryhmät, jos kyseessä on suurempi validointi, sekä heidän vastuualueensa. Roolit ovat seuraavat:

Työkalun omistaja on henkilö, joka on viime kädessä vastuussa työkalun validoinnista.

Projektimanageri on henkilö, joka hyväksyy validoinnin aikataulun ja vaadittavatresurssit, sekä työskentelee yhteistyössä työkalun omistajan kanssa mahdollisten ongelmien ilmaantuessa.

Validointimanageri on henkilö, joka on vastuussa työkalun testauksesta, validaatio- ja testi-suunnitelmien luomisesta sekä testi- ja validaatio-raporttien kirjottamisesta. Validointimanageri myös valitsee ja kouluttaa testaajat.

Tekninen asiantuntija on henkilö, joka tuntee työkalun tekniset ominaisuudet parhaiten. Hän vastuussa vaatimusten määrittelystä, testi proseduureista sekä niiden yhteyksien jäljitettävyydestä.

Laatuvastaava on henkilö, joka huolehtii, että laatujärjestelmiä ja määräyksiä noudatetaan.

Järjestelmän vaikutus

Seuraavaksi tulee arvioida työkalun vaikutusta tuotteen laatuun, jos työkalu ei jostain syystä toimisikaan niin kuin sen pitäisi. Arviossa määritellään mahdollisen vikatilanteen havaittavuus asteikolla matala tai korkea sekä vikatilanteen haittavaikutus asteikolla pieni tai suuri. Kokonaistulos saadaan matriisista (taulukko 8).

Taulukko 8: System impact

		Havaittavuus	
		Korkea	Matala
Haitta-vaikutus	Pieni	Matala	Keskiverto
	Suuri	Keskiverto	Korkea

Robot Frameworkin vikatilanteiden havaittavuudeksi määriteltiin korkea, koska testitulokset katselmoidaan ennen hyväksymistä. Käyttämällä positiivisia sekä negatiivisia testitapauksia työkalun mahdollinen vikakäyttäytyminen saadaan hyvin näkyviin koska on hyvin epätodennäköistä että ne menisivät hyväksytysti lävitse.

Haittavaikutukseksi Robot Framework sai tuloksen suuri. Koska Robot Frameworkia käytetään suoraan tuotteiden verifikaatio, testaukseen mahdollinen vikakäyttäytyminen voi johtaa havaitsemattomiin virheisiin lopputuotteessa. Kokonaistulokseksi Robot Frameworkille saatiin matriisista keskivertovaikutus.

Työkalun luokittelu

Työkalut on luokiteltu viiteen eri kategoriaan riippuen niiden monimutkaisuudesta ja siitä, kuka työkalun on luonut. Riippuen työkalun kategoriasta validaatiotarpeet vaihtelevat hieman. Kategoriat ovat seuraavat:

Kategoriaan 1 kuuluvat kaupallisesti saatavilla olevat käyttöjärjestelmät. Yleisesti käytössä olevat käyttöjärjestelmät eivät sinällään vaadi validointia, mutta käyttöjärjestelmän ominaisuudet, joita lopputuote käyttää, voidaan validoida.

Kategoriaan 2 kuuluvat älykkäät laitteet, kuten hallittavat kytkimet ja reitittimet. Tämän kategorian tuotteille on oma validointi-prosessinsa.

Kategoriaan 3 kuuluvat kaupallisesti saatavilla olevat ohjelmat, jotka ovat valmiita käyttöön ilman asetusten muokkaamista. Vaatimuksissa keskitytään määrittelemään,

kuinka työkalu täyttää käyttötarkoituksensa. Validointi voi myös käsittää työkalun asennuksen GE Healthcaren käyttöympäristöön.

Kategoriaan 4 kuuluvat kaupallisesti saatavilla olevat ohjelmat, jotka vaativat asetusten muokkaamista, jotta ne sopivat käyttötarkoitukseensa. Vaatimuksissa keskitytään määrittelemään, kuinka työkalu täyttää käyttötarkoituksensa sekä asetusten muokkauksen tarve. Validointi voi myös käsittää työkalun asennuksen GE Healthcaren käyttöympäristöön.

Kategoriaan 5 kuuluvat kaupallisesti saatavilla olevat ohjelmat, jotka vaativat jatkokehitystä, jotta ne täyttävät käyttötarkoituksensa. Kategoriaan kuuluvat myös GE Healthcaressa kehitetyt ohjelmistot, moduulit sekä skriptit. Vaatimuksissa keskitytään määrittelemään, kuinka työkalu täyttää käyttötarkoituksensa, asetusten muokkauksen tarve sekä oman ohjelmakoodin toiminta. Validointi voi myös käsittää työkalun asennuksen GE Healthcaren käyttöympäristöön.

Robot Frameworkia käytetään suoraan sellaisenaan ilman ohjelmiston tai asetusten muokkaamista, joten se sai kategoriakseen 3.

Toimittajan arviointi

Kaupallisille työkaluille on tehtävä toimittajan arviointi. Siinä selvitetään toimittajan myöntyväisyyttä työkalun kehittämiseen, kouluttamiseen ja GE-laatustandardien seuraamiseen sekä halukkuutta ja kykyä tarjota tukea. Jos toimittaja ei kykene täyttämään kaikkia vaatimuksia, se ei tarkoita, ettei työkalua voisi käyttää ollenkaan, vaan tällöin on tehtävä riskianalyysi havaituista ongelmakohdista ennen kuin validointia voidaan jatkaa.

Robot Framework on avoimen lähdekoodin ohjelma, sekä lisäksi se täyttää yrityksen käyttötarpeen suoraan, ilman ohjelman muokkausta, joten toimittajan arviointia ei tarvinnut tehdä.

Elektroniset tallenteet ja allekirjoitukset

Jos työkalu luo elektronisia tallenteita tai allekirjoituksia, on ne määriteltävä käyttäjävaatimuksissa. Käyttäjävaatimusten tulee määritellä vaadittavat toimenpiteet, jotta FDA:n vaatimukset koskien elektronisia tallenteita ja allekirjoituksia täyttyvät.

Robot Framework luo sähköisiä testitulosraportteja mutta se ei vaadi tai luo elektronisia allekirjoituksia. Tulosraportit tallennetaan, hallitaan ja hyväksytään yrityksessä käytössä olevassa dokumenttien hallintajärjestelmässä, joten niiden osalta ei tarvitse luoda käyttäjävaatimuksia.

Dokumentaatio

Validointisuunnitelmassa tulee määritellä kaikki tuotokset, jotka syntyvät validaatiosta. Tuotokset ovat yleensä dokumentaatiota, mutta jossain tapauksissa myös pieniä ohjelmia, joita tarvitaan työkalun validoimiseen. Validointisuunnitelman on myös määriteltävä, kuka tai ketkä omistavat, katselmoivat sekä hyväksyvät kunkin tuotoksen.

Vaadittaviin tuotoksiin vaikuttavat tuotteen luokitus sekä järjestelmän vaikutus -tulos. Robot Frameworkin tapauksessa vaaditaan seuraavat tuotokset:

– System Purpose and Intent (Validation Plan)
– GxP Assessment
– Supplier Assessment (Validation Plan)
– Validation Plan
– Electronic Records and Signatures Assessment (Validation Plan)
– User Requirements Specification (Requirement Specification)
– System Requirements Specification (Requirement Specification)
– System Design Specifications (Requirement Specification)
– Requirements Traceability (Test Protocol)
– System Use and Administrator Work Instructions
– Test Plan
– Test Scripts and Procedures
– Test Results

– Test Summary (Validation Report)
– Validation Report.

Kaikkia vaadittuja tuotoksia ei ole pakko luoda omiin dokumentteihinsa vaan niitä voidaan mahdollisuuksien mukaan yhdistellä, varsinkin pienimuotoisemman validaation ollessa kyseessä. Sulkumerkkien sisällä on mainittu dokumentti, josta kyseinen informaation löytyy, jos tuotokselle ei luoda omaa dokumenttia. Työkalun käyttöohjeille ei luoda omaa dokumenttia vaan riittää, että viitataan internet-osoitteeseen josta kyseinen informaatio löytyy.

Tuotosten omistajat, katselmoijat ja hyväksyjät saadaan suoraan validointiprosessin ohjeistuksesta, niitä ei tässä erikseen määritellä.

Validoinnin suunnittelu

Validoinnin suunnitteluvaiheessa on tarkoitus dokumentoida lähestymistapa validointiin, joka perustuu GxP, System Impact ja toimittajan arviointi -tuloksiin sekä työkalun luokitukseen. Suunnittelussa myös identifioidaan mahdolliset riskit, jotka liittyvät työkaluun, ja suunnitelma niiden minimoimiseksi.

Validoinnin lähestymistavasta tulee selvittää, kuinka vaatimukset verifioidaan, kuinka vaatimusten jäljitettävyys hoidetaan sekä hyväksyntäkriteria validaatiolle. Vaatimusten verifiointi tehdään yleensä testaamalla, mutta joissakin tapauksissa se ei ole mahdollista, ja silloin voidaan verifiointi suorittaa esimerkiksi koodikatselmuksella tai analyysillä. Jäljitettävyydellä tarkoitetaan vaatimusten ja testien suhdetta, eli missä testissä mikäkin vaatimus verifioidaan. Hyväksyntäkriteereillä määritellään se tilanne jolloin voidaan sanoa työkalun validoinnin olevan valmis.

Robot Frameworkin tapauksessa validoinnin lähestymistapa on tehdä kaikkien vaatimusten verifiointi testaamalla, mikäli se vain on mahdollista. Tarvittaessa turvaudutaan analysointiin tai koodikatselmointiin, jos jokin vaatimus ei ole testattavissa.

Koska Robot Frameworkia käytetään sellaisenaan ilman mitään ohjelmakoodin muokkauksia, ja koska Robot Frameworkin asennukseen ei liity monimutkaista asennusprosessia tai monimutkaista käyttöympäristöä, on riittävää verifioida ja testata vain asennetun ohjelman toiminnallisuus.

Osassa verifikaatiotestejä käytetään varta vasten luotuja HTML-, Python- ja Java-tiedostoja syötteenä työkalulle, ja ydintiimi tulee katselmoimaan nämä tiedostot ennen niiden käyttöönottoa.

Vaatimusten jäljitettävyys tehdään testidokumenttiin. Jokaisen testin alussa luetellaan niiden vaatimusten tunnukset, joita kyseinen testi verifioi. Lisäksi testidokumentin loppuun tulee yhteenvetotaulukko, josta käy ilmi, mitkä testit verifioi minkäkin vaatimuksen.

Kriteerinä Robot Frameworkin validoinnin hyväksynnälle on se vaihe, kun kaikki testit on hyväksyttävästi suoritettu sekä kaikki Validation Plan -dokumentissa mainitut tuotokset on hyväksytyssä tilassa dokumenttien hallintajärjestelmässä.

Koska kyseessä on suhteellisen pienimuotoinen validointi, yhdistettiin käyttäjävaatimukset sekä järjestelmävaatimukset yhdeksi vaatimusdokumentiksi. Tämän lisäksi kaikki vaatimukset testataan yhdessä testidokumentissa ilman erinäistä jaottelua.

Käyttöympäristön kelpoisuus ja ohjelmiston asennus

Validointisuunnitelmassa tulee olla suunnitelma työkalun käyttöympäristön kelpoisuuden varmistamisesta, etenkin jos työkalu tulee osaksi ennaltaan validoitua käyttöympäristöä. Jos työkalu vaatii toimiakseen erityisen käyttöympäristön tai toisia ohjelmia, tulee ne määritellä. Käyttöympäristön tulee olla validointitesteissä samanlainen kuin peruskäyttäjällä.

Työkalun asennuksesta tulee olla suunnitelma, josta käy ilmi mahdolliset riskit, asennusympäristön vaatimukset, skriptit sekä ongelmien hallinta. Suunnitelmasta tulee

käydä ilmi, kuinka varmistetaan, että asennusympäristö on sopiva, sekä kuinka varmistetaan asennuksen jälkeen, että asennus on suoritettu onnistuneesti.

Robot Frameworkin asennusohjeistuksessa ei määritellä käyttöympäristöä kovin tarkasti, se voidaan asentaa sekä Windows- että Linux-ympäristöön, johon on asennettu Python 2.5. Jos tahdotaan käyttää Java-kirjastoja, tulee myös Java-ajoympäristö sekä Jython 2.5 olla asennettuna. Koska Robot Framework ei tule osaksi validoitua käyttöympäristöä ja se tullaan asentamaan tarvittaessa tavalliselle työasema-tietokoneelle, sitä varten ei tarvitse luoda käyttöympäristön kelpoisuussuunnitelmaa. Robot Frameworkin validaatiossa tulemme suorittamaan kaikki testit niin Windows- kuin Linux-ympäristössä.

Järjestelmän ylläpito

Työkalulle tulisi luoda suunitelma ongelmatilanteiden varalle; kuinka varmuuskopiointi tai järjestelmän palautus hoidetaan. Robot Frameworkin käyttöluonteen takia ei suunnitelmaa tarvita. Työkalu ei tallenna mitään tietoja, ja ongelmatilanteiden sattuessa voidaan ohjelma asentaa uusiksi.

Dokumentaatio ja toimintatavat

Viimeisenä kohtana validointisuunnitelmassa on määritellä, mistä työkalu sekä mahdolliset konfiguraatio-tiedostot ja ohjeet löytyvät, mihin ja miten mahdolliset validoinnissa löydettävät poikkeamat dokumentoidaan sekä mihin kaikki validaatiossa syntyvät dokumentit tallennetaan.

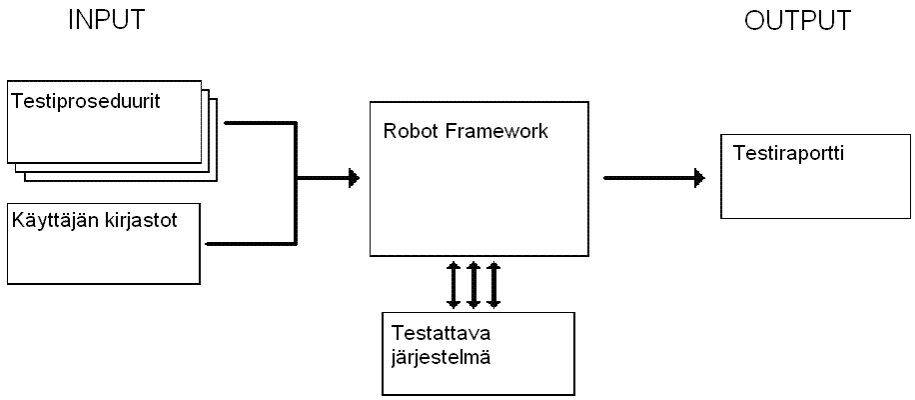
Robot Frameworkin tapauksessa, jossa omia konfiguraatio tiedostoja ei tarvita, on riittävää viitata Robot Frameworkin kotisivuille, josta ohjelma on ladattavissa. Poikkeamia varten ei luoda omaa tietokantaa yrityksen poikkeamien hallintajärjestelmään vaan poikkeamat esitellään ja käsitellään testiraportissa. Kaikki validointi dokumentaatio säilytetään yrityksen dokumenttien hallintajärjestelmässä, MyWorkshopissa.

4.5 Vaatimukset

Kun validointisuunnitelma on valmis, voidaan seuraavaksi siirtyä määrittelemään vaatimuksia, jotka työkalun tulee täyttää, jotta se voidaan ottaa käyttöön. Työkalulle tulee määritellä sekä käyttäjä- että järjestelmävaatimukset.

Käyttäjävaatimukset voivat olla käyttöskenaario muodossa tai sitten yksittäisinä vaatimuksina. Vaatimusten tulee kuvata selvästi, mitä käyttäjä haluaa työkalun tekevän, miten työkalua käytetään ja mitä dataa työkalu tarvitsee toimiakseen. Järjestelmävaatimukset ovat tarkempia kuin käyttäjävaatimukset, ne kertovat yleensä, kuinka käyttäjävaatimukset teknisesti toteutetaan ja kuinka tuote täyttää käyttötarkoituksensa. Kaikkien vaatimusten tulee olla testattavissa tai muilla keinoin verifioitavissa. Kaikki vaatimukset tulee myös jollain tavoin linkittää niitä verifioiviin testeihin, jotta jäljitettävyys säilyy.

Robot Frameworkin vaatimusten luonti aloitettiin ensiksi luomalla hyvin yksinkertainen järjestelmädiagrammi josta käy selville miten työkalu toimii. Järjestelmä diagrammi esitellään kuviossa 1.



Kuvio 1. Robot Frameworkin järjestelmädiagrammi.

Robot Framework ottaa sisäänsä testitapauksia sekä käyttäjän omia kirjastoja. Tämän jälkeen työkalu on vuorovaikutuksissa testattavan järjestelmän kanssa ja lopputuloksena saadaan testiraportti.

Robot Frameworkin validoinnin pienuuden vuoksi käyttäjä- ja järjestelmävaatimuksia ei eritelty omiin dokumentteihinsa vaan luotiin yksi vaatimusdokumentti, joka sisältää vaatimuksia, jotka ovat luonteeltaan molempia vaatimustyyppejä. Seuraavissa kappaleissa on esitelty vaatimukset, jaoteltuna omiin kategorioihinsa.

Asennus ja konfiguraatio

1. Työkalulle tulee olla Linux sekä Windows asennuspaketit.
2. Työkalulla tulee olla mahdollista käynnistää useamman testitiedoston ajo kerralla.
3. Työkalulla tulee voida käynnistää testitiedostojen ajoja hakemistojen mukaan.
4. Työkalulla tulee voida käynnistää testejä tagien perusteella.
5. Työkalulla tulee olla mahdollista antaa testiajolle muuttujia komentokehotteesta tai tekstitiedostosta.

Testien muotoilu

1. Testitapaukset tulee määritellä HTML muodossa. Testit ovat eroteltu omiin tauluihinsa.
2. Työkalun tulee tukea käyttäjän omia Python- ja Java-kirjastoja.

Testien ajo

1. Työkalun tulee ilmoittaa testiajon tulokseksi PASS, jos kaikki testin avainsanat suoritetaan onnistuneesti, pois lukien ne avainsanat, joiden odotetaankin epäonnistuvan (negatiivinen testaus).
2. Työkalun tulee ilmoittaa testiajon tulokseksi FAIL, jos yksikin testin avainsanoista suoritetaan epäonnistuneesti, pois lukien ne avainsanat, joiden odotetaan epäonnistuvan (negatiivinen testaus).
3. Testiajon aikana tulee olla mahdollista saada arvo tai valinta käyttäjältä; testiajo pysäytetään, kunnes arvo tai valinta on syötetty.
4. Testiajon aikana tulee olla mahdollista saada PASS- tai FAIL-päätös käyttäjältä, testiajo pysäytetään, kunnes valinta on syötetty.

5. Käyttäjien omissa kirjastoissa olevien avainsanojen tulee voida ottaa sisäänsä argumentteja sekä niiden tulee voida antaa palautusarvoja.
6. Työkalun tulee merkata testin tulokseksi FAIL jos käyttäjien omissa kirjastoissa tapahtuu exception.

Testitulosten raportointi

1. Valmiin testiajon jälkeen tulee työkalun luoda HTML muodossa oleva testiraportti josta löytyy testiajon yhteenveto.
2. Valmiin testiajon jälkeen tulee työkalun luoda loki tiedosto josta löytyy jokainen testitapahtuma askel askeleelta.
3. Keskeytyneen testiajon jälkeen testiraporttia ei tule luoda.

Versiointi

1. Työkalun versiotieto tulee voida selvittää työkalusta.

4.6 Testisuunnitelma

Kun työkalulle on saatu määriteltyä vaatimukset, voidaan niiden pohjalta alkaa suunnitella testausta. Testisuunnitelmassa tulee määritellä testauksen aloitusehdot, testimenetelmät, testauksen läpimenon kriteerit sekä testaukseen vaadittava koulutus. Testisuunnitelmassa määritellään ajettavat testiprotokollat ja niiden ajoympäristöt sekä myös mahdolliset muut testausmenetelmät.

Robot Frameworkin tapauksessa testauksen aloitusehdoiksi määriteltiin, että seuraavat dokumentit tulee olla hyväksyttynä dokumenttien versionhallinnassa: Validation Plan, Requirement Specifications, Test Plan, Test Procedures ja Test files. Testimenetelmiksi määriteltiin, että vaatimukset verifioidaan testaamalla, tämän lisäksi testataan kaikki Robot Frameworkin sisäänrakennetut avainsanat sekä katselmoidaan Robot Frameworkin omasta tietokannasta kaikki avoimet havainnot ja anlysoidaan, onko niillä vaikutusta siihen miten työkalua tullaan käyttämään. Jotta testaus voidaan katsoa suoritetuksi, tulee kaikkien suunniteltujen testien olla suoritettu, testauksessa nousseiden havintojen olla käsitelty sekä testiraportin tulee olla valmis ja hyväksytty

dokumenttien versionhallinnassa. Robot Frameworkin testien suorittaminen ei vaadi mitään erikoista koulutusta, tästä johtuen koulutustarpeiksi määriteltiin vain vaatimuksiin ja testeihin tutustuminen ennen testien suorittamista.

Robot Frameworkin testisuunnitelmassa määrittelemme että tulemme ajamaan testiproseduurit kahteen kertaan, sekä Windows XP:llä että Ubuntu 10.4 LTS:llä. Testisuunnitelma kertoo myös kuinka Robot Frameworkin omassa tietokannassa olevat avoimet havainnot tulee analysoida ja käsitellä.

4.7 Testiproseduuridokumentti

Testiproseduureilla varmistetaan, että työkalu täyttää sille asetetut vaatimukset. Testiproseduurissa tulee olla testiympäristön asennusohjeet ja mahdollisesti testi jolla varmistetaan että testiympäristö on oikein asennettu. Testien tulee kuvata selkeästi mitä aktiviteetteja testaajan tulee suorittaa ja mikä on odotettu tulos. Testitapauksissa tulee olla selkeästi merkattu alue johon testaaja voi kirjoittaa testin tuloksen sekä kohta johon testaaja merkitsee onko testi mennyt läpi vai epäonnistunut.

Robot Frameworkin testiproseduurit ovat pääasiassa ennalta luotuja testitiedostoja, jotka testaaja ajaa komentoriviltä. Tämän jälkeen testaaja kopioi joko komentorivillä tai raporttitiedostossa näkyvän tulosteen testidokumenttiin ja tekee päätöksen, onko testi mennyt läpi vai ei. Muutama testiproseduuri ei käytä ennalta määriteltyjä testitiedostoja vaan testaaja kirjoittaa käskyn itse komentoriville ja ajaa sen. Osa testeistä taas pyytää testaaja tarkistamaan, että määritellyt testiraporttitiedostot on luotu. Kaikki testeissä käytettävät testitiedostot katselmoidaan ja hyväksytään yhdessä testiproseduuridokumentin kanssa.

4.7.1 Sisäänrakennettujen avainsanojen verifiointi

Pääosa luoduista testeistä on sisäänrakennettujen avainsanojen verifiointia. Yhteensä verifioituja avainsanoja on 81. Kaikille avainsanoille luodaan testit, jotka varmistavat, että ne toimivat, kuten manuaalissa kuvataan. Avainsanan luonteesta riippuen jokaiselle avainsanalle luodaan testi, joka palauttaa tuloksen PASS, sekä testi joka

palauttaa tuloksen FAIL. Eräät avainsanat tekevät poikkeuksen tähän, koska ne eivät palauta kuin yhden tuloksen, joko PASS tai FAIL, esimerkiksi avainsana "Fail" palauttaa vain tuloksen FAIL eikä koskaan PASS.

Koska sisäänrakennettujen avainsanojen verifiointi tahdottiin pääosin automatisoida, tulee aivan ensimmäiseksi manuaalisesti verifioida muutama avainsana, joiden avulla voidaan tämän jälkeen automaattitestauksella verifioida loput avainsanat. Tärkeimmät manuaalisesti verifioitavat avainsanat ovat "Run Keyword and Expect Error", jolla ajetaan varsinainen avainsana ja odotetaan tulosta FAIL, sekä avainsanat, joilla tehdään muuttujien vertailua tai argumenttien evaluointia, kuten ”Should Be Equal”, ”Should Be True” ja ”Should Not Be True”.

Taulukossa 9 on avainsanan "Should Be Equal As Integers" verifiointitesti, jossa ensimmäisenä annetaan avainsanalle argumentteina kaksi samaa lukua ja odotetaan vastausta PASS. Kun avainsanalle annetaan vain yksi argumentti, argumentti, joka ei ole luku, tai kaksi toisistaan eroavaa lukua, avainsana palauttaa tuloksen FAIL.

Taulukko 9: "Should Be Equal As Integers" avainsanan verifiointitesti.

Should Be Equal As Integers	5	5		
Run Keyword And Expect Error	*	Should Be Equal As Integers	5	
Run Keyword And Expect Error	*	Should Be Equal As Integers	5	A
Run Keyword And Expect Error	*	Should Be Equal As Integers	5	4

Edellä olevan testin tulisi mennä läpi kokonaistuloksella PASS, koska avainsana "Run Keyword And Expect Error" odottaa, että varsinainen avainsana "Should Be Equal As Integers" palauttaa tuloksen FAIL.

4.7.2 Testikirjastotuki

Koska Robot Framework tukee Python- ja Java-kirjastoja tulee näiden tuki myös testata. Molempia ohjelmointikieliä varten luotiin yksinkertainen kirjasto, jolla testataan, että kirjaston sisältämää avainsanaa voidaan kutsua, avainsanalle voidaan

antaa argumentteja sekä avainsana voi palauttaa arvon. Lisäksi testataan, että poikkeus kirjastossa aiheuttaa avainsanan tulokseksi arvon FAIL.

Esimerkkikoodissa 1 on listaus Python-tuen testausta varten luodusta tiedostosta. Python-tiedosto otetaan käyttöön Robot Frameworkissä määrittelemällä se testin alustusosiossa, tämän jälkeen Python-tiedostossa olevia funktioita voidaan käyttää avainsanoina Robot Frameworkin testeissä.

```
class python_support:

    """Library for testing python support in RF. """

    def python_support_test(self):

        return 1

    def python_argument_test(self, somevalue):

        somevalue = int(somevalue)
        return somevalue*2

    def python_exception_test(self):

        raise Exception ("Exception has raised in python code,
step should be fail")
```

Esimerkkikoodi 1. Python-tuki.

Kutsumalla avainsanaa ”python support test” Robot Frameworkista verifioidaan, että Python-kirjastojen käyttö on mahdollista. Kutsumalla avainsanaa ”python argument test” ja antamalla sille argumenttina jokin luku, verifioidaan, että on mahdollista antaa avainsanalle argumentti sekä avainsana voi palauttaa arvon. Validointitestissä kutsutaan avainsanaa ”python argument test” argumentilla 2 ja avainsanan palautusarvo tallennetaan muuttujaan. Tämän jälkeen muuttujan arvoa verrataan lukuun 4. Jos vertaus on tosi, voidaan olla varmoja, että python-funktio sai argumenttina annetun arvon ja osasi palauttaa oikean arvon.

Poikkeama verifioidaan avainsanalla ”python exception test”, siinä aiheutetaan poikkeama palautusarvolla ”Exception has raised in python code, step should be fail”. Validointitestissä kutsutaan varsinaista avainsanaa ”python exception test” avainsanan ”Run Keyword And Expect Error” avulla, jolloin validointi on onnistunut, jos varsinainen

avainsana saa tuloksen FAIL ja se palauttaa python-kirjastossa olevan lauseen ”Exception has raised in python code, step should be fail”.

Java-kirjaston testaus on hyvin samanlainen kuin Pythoninkin. Testit ovat toimintaperiaatteeltaan lähes identtiset, ainoastaan avainsanat, poikkeaman palautusarvo sekä ohjelmointikieli eroavat. Ohessa on listaus Java-tuen testauksessa käytettävän tiedoston sisällöstä (esimerkkikoodi 2).

```
public class java_lib {

    public static int JavaSupportTest() {
        return 1;
    }

    public static int JavaArgumentTest(int somevalue) {
        return somevalue*2;
    }

    public static int JavaExceptionTest() {
        throw new java.lang.RuntimeException("Exception was
thrown in java code, step should be fail");
    }

}
```

Esimerkkikoodi 2. Java-tuki.

4.8 Testiajo

Testiajo voi alkaa, kun kaikki testisuunnitelmassa olevat testin aloitusehdot täyttyvät. Testit tulee ajaa sellainen henkilö, joka ei ole ollut mukana luomassa testejä. Testit suoritetaan testiproseduuridokumenttia seuraten ja tulokset, allekirjoitus sekä päiväys kirjataan dokumenttiin. Työkalun validoinnista vastaava testausmanageri varaa tarvittavat testaajat hyvissä ajoin ja varmistaa, että mahdollinen koulutus suoritetaan ennen testien alkua.

Robot Frameworkin testauksen suoritti yksi henkilö ja testimanageri avusti mahdollisissa ongelmatilanteissa. Testiajoja oli kaksi, Windows- ja Linux-käyttöjärjestelmille. Kummastakaan testiajosta ei löytynyt yhtään havaintoa vaan kaikki testit menivät hyväksytysti läpi. Testiproseduurien suorittamisen jälkeen tekninen asiantuntija kävi läpi kaikki avoimet havainnot Robot Frameworkin

havaintotietokannasta. Havaintotietokannassa oli katselmointihetkellä kahdeksan avointa havaintoa, mutta yksikään niistä ei analysoinnin perusteella vaikuta haittaavasti työkalun käyttöön.

4.9 Testiraportti

Testiajon jälkeen tulee luoda testiraportti, jossa käydään läpi testisuoritus ja varmistetaan, että kaikki testisuunnitelmassa määritellyt aktiviteetit on suoritettu. Raportissa listataan kaikki havaitut havainnot ja niiden käsittelyn tulos sekä mahdolliset poikkeamat testisuunnitelmasta, jos alkuperäisestä suunnitelmasta on poikettu.

Robot Frameworkin validoinnissa testiraportille ei luotu omaa dokumenttia vaan se liitettiin osaksi validointiraportti dokumenttia. Testiajojen tulokset sekä avoimien havaintojen analyysin tulokset kirjattiin testiraporttiin.

4.10 Validointiraportti

Validointiraportissa tulee esittää yhteenveto koko validaatioprosessista. Validointiraportissa tulee olla dokumentti referenssit kaikkiin validointisuunnitelmassa esitettyihin tuotoksiin, yhteenveto kuinka asetettu hyväksyntäkriteria kohdattiin, mahdolliset poikkeamat validointisuunnitelmasta sekä validoinnin lopputulos. Validointiraportista tulee käydä ilmi, mikä työkalun versio ja käyttöympäristö validoitiin. Kun validointiraportti on katselmoitu ja hyväksytty dokumenttien versionhallintaan, validoinnin katsotaan olevan loppuunsuoritettu.

Robot Frameworkin validointiraportista käy ilmi, että validointi suoritettiin versiolle 2.5 sekä Windows XP- että Ubuntu 10.4 LTS -käyttöympäristöissä. Validointisuunnitelmasta poikettiin testitulosten hyväksyjien suhteen, alkuperäisessä suunnitelmassa ei määritelty tuloksille ketään hyväksyjää, mutta testitulokset hyväksyi sekä testimanageri että testin suorittaja. Validointiraportista käy ilmi, että kaikki validointisuunnitelmassa määritetyt hyväksyntäkriteerit on täytetty, validoinnin lopputulos on hyväksytty ja työkalua voi käyttää GE Healthcaren kehitys- ja verifikaatiotestauksessa.

5 Lopuksi

Insinöörityössä saavutettiin tavoite, eli Robot Framework saatiin validoitua onnistuneesti ja työkalua voidaan nyt käyttää yrityksessä. Validointiprosessin aikana kävi selväksi, että se on hieman raskas pienten ohjelmien hyväksyntään ja on alun perin suunniteltu pääasiassa monimutkaisempien työkalujen ja järjestelmien hyväksyntään.

Validointi suoritettiin Robot Framework versiolle 2.5, ja tämän jälkeen on työkalusta julkaistu jo viisi uudempaa versiota (7.1.2011). Uuden version validointi on kuitenkin huomattavasti kevyempi prosessi, kun jo olemassa olevaa dokumentaatiota voidaan käyttää hyväksi.

Insinöörityötä kirjoitettaessa on Mobile Viewers -projekti jo käyttänyt Robot Frameworkia monitorirajapinnan testaamiseen, ja sillä saavutettiin huomattavat edut verrattuna manuaaliseen testaukseen. Manuaalinen testaus olisi sitonut yhden testaajan viikoksi, kun automaattinen testaus tarvitsi huomiota vain yhden tunnin verran. Työkalulla ajettiin myös epävirallisia automaatiotestejä pitkin kehitysprosessin, jotka paljastivat epäkohtia tuotteesta. Nämä epäkohdat olisi normaalisti löydetty vasta tuotteen lopullisessa hyväksyntätestissä ja aikataulu olisi venynyt.

Lähteet

1 GE Healthcare. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/GE_Healthcare>. Luettu 28.4.2011.

2 Robot Framework Quick Start Guide. 2009. Verkkodokumentti. Google. <http://robotframework.googlecode.com/hg/doc/quickstart/quickstart.html>. Luettu 28.4.2011.

3 Overview of Device Regulation. Verkkodokumentti. U.S. Food and Drug Administration. <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/default.htm>. Luettu 3.10.2010.

4 Quality System Regulation. Verkkodokumentti. U.S. Food and Drug Administration. <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/QualitySystemsRegulations/default.htm>. Luettu 3.10.2010.