TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietokonetekniikan suuntautumisvaihtoehto

Toni Imppu

SERIES 60 -MATKAPUHELIMEN JÄRJESTELMÄTESTAUKSEN AUTOMATISOINTI TEST-X-TYÖKALULLA

Insinöörityö, joka on jätetty opinnäytteenä tarkastettavaksi insinöörin tutkintoa varten
Tampereella 18.10.2005.

Työn valvoja: Yliopettaja Kai Poutanen
Työn ohjaaja: Site Manager FM Tero Jokinen

Tekijä: Toni Imppu
Työn nimi: Series 60 -matkapuhelimen järjestelmätestauksen automatisointi Test-X-työkalulla
Päivämäärä: 18.10.2005
Sivumäärä: 29
Hakusanat: Järjestelmätestaus, Series 60, testauksen automatisointi
Koulutusohjelma: Tietotekniikka
Suuntautumisvaihtoehto: Tietokonetekniikka

Työn valvoja: Yliopettaja Kai Poutanen

Työn ohjaaja: Site Manager FM Tero Jokinen
Tieto-X Oyj, Tampere

Testauksella on merkittävä tehtävä matkapuhelimien tuotekehityksessä. Sillä pyritään varmistamaan tuotteen toimivuus. Matkapuhelinten jatkuvasti lisääntyvät ominaisuudet lisäävät testauksen tarvetta ja sen hankaluutta. Siksi testauksen automatisointiin panostetaan kovasti. Tässä työssä tutkitaan Series 60 -matkapuhelimen järjestelmätestausta ja sen automatisoimista.

Testauksen automatisointityökaluja on markkinoilla jonkin verran. Näiden työkalujen toimivuudessa ja käytettävyydessä esiintyy useasti ongelmia. Toki myös hyviä tuotteita on, mutta koskaan ei voi tietää, soveltuuko työkalu juuri siihen käyttötarkoitukseen, johon sitä etsitään. Osittain tuosta syystä Tieto-X Oyj teki oman työkalun, nimeltä Test-X, Series 60 -matkapuhelimien järjestelmätestauksen automatisointiin. Tässä työssä tutustutaan tähän työkaluun ja sen ominaisuuksiin.

Aluksi tutkittiin, mitä testejä regressiotestisetistä voidaan automatisoida Test-X-työkalulla. Työn tuloksena Test-X-työkalulla saatiin automatisoitua yli puolet regressiotesteistä. Samalla myös rasitus- ja suorituskykytestejä automatisoitiin. Myös testaukseen käytettävä aika väheni, ja testaustyö helpottui paljonkin jossain tapauksissa. Kehitystyötä kuitenkin tarvitaan vielä paljon, jotta testausta saadaan automatisoitua enemmän ja työkalua kehitettyä paremmaksi.

Author:	Toni Imppu
Name of the thesis:	Automating system testing with Test-X tool in Series 60 environment
Date:	18.10.2005
Pages:	29
Keywords:	Software testing, system testing, Series 60
Degree programme:	Computer systems engineering
Specialization:	Computer engineering
Supervisor:	Principal Lecturer Kai Poutanen
Instructor:	Site Manager FM Tero Jokinen Tieto-X Oyj, Tampere

Software testing is nowadays a very important issue in the mobile phones development. With testing we ensure that the product works correctly. More functions are coming to the mobile phones all the time and those are increasing the load and the difficulties of testing. That is why corporations are investing in test automation. In this thesis is researched the system testing of the Series 60 platform mobile phones and how to automate that testing.

There are some automating test tools available on the market. Usually there are problems with the tool's functionality and usability. Of course, there are also good tools available but you never know is the tool right for the purpose that it is searched for. That is one reason why Tieto-X Oyj made an own automating test tool for Series 60 platform mobile phones. The tool is called Test-X and it is made for system testing. Test-X tool and functions of the tool are introduced in this thesis.

First are studied the test cases in regression test set that can be automated with Test-X and so save time of testing. With this free time a tester can do manual testing and develop new test cases or improve existing ones.

Result of this thesis was that quite many regression test cases managed to automate with Test-X. Testing time was decreased and testing work got much easier with some cases. But a lot of development is still needed for the Test-X tool and for the automated test cases.

ALKUSANAT

Tämä insinöörityö on tehty Tieto-X Oyj:n tietoliikenneyksikössä vuoden 2005 toukokuun ja syyskuun välisenä aikana. Työn ohjaajana on toiminut FM Tero Jokinen Tieto-X Oyj:stä. Työn valvojana Tampereen ammattikorkeakoulussa toimi yliopettaja Kai Poutanen.

Kiitän Tieto-X:n Tampereen Site Manageria Tero Jokista, joka antoi mielenkiintoisen aiheen insinöörityölleni. Lisäksi haluan kiittää työprojektiini kuuluvia henkilöitä, joilta olen saanut neuvoja insinöörityöhöni.

Lopuksi haluan erityisesti kiittää avovaimoani Hanna-Maria kaikesta tuesta ja kannustuksesta.

Tampereella 18. lokakuuta 2005

_____

Toni Imppu

SISÄLLYSLUETTELO

KÄYTETYT LYHENTEET JA TERMIT

3G	Third Generation. Yleinen lyhenne ns. ”kolmannen sukupolven” matkapuhelinteknologioille.
Bluetooth	Langaton tiedonsiirtoyhteys.
C++	Oliopohjainen ohjelmointikieli.
GPRS	General Packet Radio Service. Tiedonsiirtotapa langattomille sovelluksille. Pakettikytkentäinen datapalvelu.
IMAP	Internet Message Access Protocol. Protokolla sähköpostien lukemiseen ja hallinnointiin.
IrDA	Infrared Data Association. Infrapunavaloon perustuva langaton tiedonsiirtotekniikka.
Java	Olioperusteinen ohjelmointikieli.
Kernel	Käyttöjärjestelmän ydin. Huolehtii tietoliikenteestä käyttäjän ja koneen välillä.
Microsoft Outlook	Tietokoneeseen saatava henkilökohtaisten tietojenhallinta- ja viestintäohjelma.
Mini SD	Mini Secure Digital. Puhelimeen/kameraan saatava muistikortti.
MMC	Multimedia Card. Puhelimeen saatava muistikortti.
MMS	Multimedia Messaging Service. Multimedian lähetykseen tarkoitettu viestipalvelu matkapuhelinympäristössä.
Moduuli	Pienin testattavissa oleva ohjelmiston osa. Moduuli on yleensä 100 - 1000 koodiriviä.
MP3	MPEG Audio Layer 3. Äänitiedosto, joka on pakattu.
PC	Personal computer. Tietokoneen yleisnimitys.
RAM	Random Access Memory. Käyttömuisti.
RealOne Player	Multimediasoitin. Voidaan esimerkiksi katsoa videoita tai kuunnella musiikkia.
Series 60	Nokian kehittämä matkapuhelinohjelmistoalusta.

SIS	Symbian Installation System, Software Installation – paketti, joka mahdollistaa sovelluksen asennuksen Symbian-matkapuhelimeen.
SMS	Short Message Service. Tekstiviestipalvelu matkapuhelinympäristössä.
Symbian Ltd	Useiden matkapuhelinvalmistajien omistama yritys, joka on erikoistunut matkapuhelimienkäyttöjärjestelmiin.
Symbian OS	Symbian Operating System. Symbianin kehittämä mobiilikäyttöjärjestelmä ja ohjelmointiympäristö.
SyncML	Protokolla datan synkronointiin käyttäjän datan ja tallennetun datan välillä.
TCP/IP	Transmission Control Protocol/Internet Protocol. Tiedonsiirtoprotokolla.
UI	User Interface. Sovelluksen käyttöliittymä.
USB	Universal Serial Bus. Tietokoneiden oheislaitteiden käyttöönottoa helpottamaan ja monipuolistamaan luotu sarjamuotoinen väylä.
XML	Extensible Markup Language. Metakieli, jolla määritellään rakenteellisia merkkauskieliä.
WAP	Wireless Application Protocol. Langattomissa tietoverkoissa käytettävä tiedonsiirtoprotokolla.

1 JOHDANTO

Matkapuhelimien kehittyminen on tuonut haasteita puhelimien valmistajille rakentaa entistä hienompia käyttöliittymiä sekä uusia ja parempia ohjelmia, jotka sopeutuvat käyttäjien vaatimuksiin. Ensin matkapuhelimella soitettiin tavallisia puheluita, mutta nykyään on mahdollista soittaa myös videopuheluita. Matkapuhelinta voidaan jo sanoa melkein tietokoneeksi, koska sillä voidaan tehdä samoja päivittäisiä asioita kuin tietokoneella.

Lisääntynyt toiminnallisuus ja uudet ominaisuudet käyttöliittymässä ovat lisänneet haasteita matkapuhelinten testauksessa. Erilaisten komponenttien integroiminen puhelimeen on luonut monimutkaisen ympäristön, jossa kaikki osat pitää saada toimimaan yhteen. Tämä on väistämättä johtanut siihen, että virheiden todennäköisyys on kasvanut. Tämän takia ohjelmistojen testaukseen on jouduttu lisäämään resursseja, jotta kaikki vaatimukset saadaan testattua. Kun testattavien asioiden määrä on kasvanut koko ajan ja henkilöstöresursseja ei voida loputtomiin kasvattaa, apuun on otettu testausautomaatio.

Testauksen suurin ongelma onkin ohjelmistojen koon nopea kasvu. Kun ohjelmoijille on luotu kehittyneitä ohjelmointiympäristöjä, joilla voidaan tuottaa yhä monimutkaisempia ohjelmistoja yhä nopeammin, ei testaamista ole kehitetty vastaavassa suhteessa. Testausmenetelmät ovat kyllä vakiintuneet, mutta testauksen helpottamiseksi tai sen nopeuttamiseksi ei ole kehitetty vastaavia apuvälineitä kuin ohjelmointiin. Samalla testattavan materiaalin määrä ja testauksen vaikeus ovat moninkertaistuneet [1]. Varsinkin graafisten käyttöliittymien käyttöönotto on lisännyt testattavan materiaalin määrää huomattavasti, koska nyt joudutaan testaamaan sekä itse ohjelmiston varsinainen toiminnallisuus että usein hyvin monimutkaisen käyttöliittymän toiminnallisuus.

Samalla yritysten ohjelmistoprojektien vaatimukset vain kovenevat: lisää ominaisuuksia, uusia ohjelmia, nopeammin ja laadukkaammin. Tuotteet pitäisi saada valmiiksi mahdollisimman nopeasti, mutta laatu ei saisi kärsiä, koska laatu vaikuttaa koko yrityksen menestykseen. Yritysten vaatimukset kyllä ymmärtää, koska kilpailu markkinoilla on kovaa.

Tässä insinöörityössä esitetään testauksen ja testauksen automatisoinnin periaatteita yleisesti sekä järjestelmätestauksen automatisointia Series 60 -ympäristössä Test-X-työkalulla. Luvussa 2 kerrotaan lähemmin Series 60 -ohjelmistoalustasta ja sen ominaisuuksista. Luvussa 3 käsitellään testauksen kulkua ja testauksen määritelmiä. Luvussa 4 tutustutaan testauksen automatisointiin, sen hyötyihin ja vaikeuksiin. Luvussa 5 esitellään Test-X-työkalua sekä Series 60 -matkapuhelimen järjestelmätestauksen automatisointia Test-X-työkalulla.

Tämän insinöörityön tavoite oli tutkia, mitkä kaikki testit, joita käytetään regressiotestauksessa, olisi mahdollista automatisoida Test-X:llä.

2 SERIES 60 -KEHITYSYMPÄRISTÖ

Matkapuhelimet ovat kehittyneet nopeasti viimeisten vuosien aikana. Jatkuvassa kehityksessä Nokia on pysynyt hyvin mukana ja kehittänyt alaa. Matkapuhelinten kehityksen suuntaa antavana yhtiönä Nokia on vahvasti mukana Series 60 -yhteisössä, joka on kehittänyt Series 60 -ohjelmistoalustan. Series 60 -ohjelmistoalusta on tarkoitettu käytettäväksi niin sanotuissa älypuhelimissa.

Nokian matkapuhelinten vahvuutena on aina ollut niiden helppokäyttöisyys, eikä Series 60 tee tässä poikkeusta. Puhelinten selvät graafiset ikonit johdattavat käyttäjän hänen haluamaansa toimintoon. Käyttöjärjestelmä on erittäin selkeä ja looginen. Uusia sovelluksia on kehityksen myötä tullut jatkuvasti lisää, mutta ne eivät ole tehneet puhelimesta monimutkaista käyttää.

Series 60 -ohjelmistoalusta on kehitetty olemaan avoin kaikille innovaatioille. Tarkoituksena on antaa kaikille mahdollisuus kehittää ja luoda mahdollisesti uusia tulevaisuuden vaatimuksia vastaavia ratkaisuja. Series 60 -ohjelmistoalustasta julkaistiin kesällä 2005 uusi, kolmas versio, johon on tullut lisää ominaisuuksia. Tarkoituksena on yhdistää Series 60 -ohjelmistoalustaan vuoden 2006 aikana Series 90 -ohjelmistoalustan ominaisuudet eli kehitystyötä tehdään koko ajan.

Jatkuvan kehityksen turvaamiseksi Nokia on tuonut Series 60 -ohjelmistoalustan kaikkien saataville. Matkapuhelinvalmistajien on mahdollisuus lisensoida kyseinen tuote ja saada oikeudet koodin muuttamiseen. Tähän mennessä matkapuhelinvalmistajista Siemens, Panasonic, Samsung, Lenovo ja LG Electronics ovat saaneet lisenssin tuotteeseen. Näistä valmistajista kaikki muut paitsi LG Electronics ovat jo julkistaneet Series 60:n pohjalle perustuvan matkapuhelimen. Kaikkiaan Series 60 -ohjelmistoalustaisia puhelimia on tullut markkinoille jo lähes 30 ja lisää tulee koko ajan. [3]

Series 60:n tarkoituksena on antaa lähdekoodi tietyin rajoituksin lisenssin haltijoiden käyttöön. Lisenssin haltijat voivat muokata ja muuttaa koodia omien projektiensa tarpeiden mukaisesti. He voivat siis tehdä mahdollisesti uusia sovelluksia mutta myös poistaa niitä omien matkapuhelinsovelluksiensa tarpeita varten. Tähän tietysti sisältyy monenlaisia rajoituksia. [2]

2.1 Toiminnalliset ominaisuudet

Series 60 -matkapuhelimet on suunniteltu niin sanotuiksi ”one-hand”-puhelimiksi [2]. Tämä tarkoittaa sitä, että puhelinta pitää pystyä hallitsemaan ainoastaan yhdellä kädellä. Tietenkin kahden käden käyttö on mahdollista. Yhdellä kädellä pystytään kuitenkin hallitsemaan kaikki toiminnot.

Kamerasovelluksen tuleminen matkapuhelimeen on antanut täysin uuden merkityksen matkapuhelimen käytölle. Kameroiden tarkkuus ja zoomaus-ominaisuudet kehittyvät koko ajan ja jo nykyisillä kameroilla saadaan hyvälaatuisia kuvia. Myös videokuvaus onnistuu Series 60 -matkapuhelimilla ja samalla tavalla videokuvauksen tarkkuus kehittyy kuten kamerankin. RealOne Playerillä voidaan soittaa musiikkia stereona tai sitten voidaan kuunnella FM-radiota. Matkapuhelin tukee muun muassa MP3-formaattia, ja siksi voidaan todeta, että matkapuhelimesta

on tullut todellinen multimedialaite. Kuvia, videokuvaa ja musiikkia voidaan lähettää ja ottaa vastaan MMS:n tai sähköpostin välityksellä.

Series 60 -matkapuhelimien suunnittelun perustana on pidetty helppokäyttöisyyttä. Periaatteena on ollut, että kaikki, jotka osaavat käyttää matkapuhelinta, osaavat käyttää myös Series 60 -laitetta [2]. Tämä pitääkin hyvin paikkansa. Graafinen käyttöliittymä luo visuaalisesti hienon näköisen käyttöliittymäkokonaisuuden, jonka käytössä ei ole vaikeuksia. Kaikki sovellukset on varustettu ikoneilla, jotka kätkevät taakseen sovelluksen toiminnot. Toimintojen välillä pystytään selailemaan viisisuuntaisella navigointinäppäimellä. Näppäin on erittäin tehokas muissakin toiminnoissa, kuten esimerkiksi WAP-sivujen selailussa. Kuvassa 1 on esitetty näkymä Nokia 6680 -matkapuhelimen käyttöliittymänäkymästä.

Kuva 1. Nokia 6680:n käyttöliittymänäkymä

Series 60 -matkapuhelimen käyttöliittymänäkymää on erittäin helppo personalisoida itselleen sopivaksi. Kuvassa 2 on asennettu puhelimeen teema, joka muuttaa käyttöliittymänäkymää toisenlaiseksi. Valmiita teemoja löytyy Internetistä paljon, joskin sellaisen voi myös itse tehdä. Taustakuvaksi voi myös asettaa puhelimella otetun kuvan.

Kuva 2. Nokia 6680:n käyttöliittymänäkymä teemalla

Myös operaattoreilla on mahdollisuus personalisoida käyttöliittymänäkymää omanlaisekseen. Operaattoreilla voi olla myös oma valikko, josta saa esimerkiksi ladattua uusia ohjelmia, pelejä tai teemoja puhelimeen helposti.

Series 60 -ohjelmistoalustapaketti on erittäin kattava ominaisuuksiltaan. Sen mukana tulevat esimerkiksi seuraavat henkilökohtaisten tietojen hallintaan liittyvät sovellukset: kalenteri, tehtävälista, kello, muistivihko ja puhelinluettelo. Näiden sovelluksien tiedot voidaan synkronoida vaikka PC:n Microsoft Outlookin kanssa.

Series 60 -paketin mukana tulevat myös kaikki puhelusovellukset, kuten myös viestisovellukset eli SMS, MMS ja sähköposti. Kun matkapuhelimessa on jo paljon henkilökohtaista tietoa, myös turvallisuuteen täytyy kiinnittää huomiota. Niinpä uusimmassa Series 60 -versiossa ohjelmille määritetään niiden oikeudet. Jos ohjelma tarvitsee paremman tason oikeuksia, se vaatii ohjelman sertifiointia. Tämä järjestelmä antaa siten myös suojaa viruksia vastaan. Seuraavassa kerrotaan hieman tarkemmin muutamasta Series 60:n tarjoamasta ominaisuudesta. [3]

2.2 Tiedonsiirto

Uusia sovelluksia Series 60 -matkapuhelimiin voidaan ladata datakaapelin, IrDA:n, Bluetoothin tai mahdollisesti muistikortin eli MMC:n avulla. Yksi vaihtoehto on hakea ja asentaa sovellukset suoraan Internetistä. Muistikortin käyttömahdollisuus on osoittautunut hyväksi tavaksi lisätä puhelimen muistikapasiteettia. Varsinkin multimediasovelluksissa tämä lisämuisti on erittäin tarpeellista.

2.2.1 Datakaapeli

Datakaapelin avulla on helppo siirtää tiedostoja tietokoneen ja matkapuhelimen välillä. Useimmiten kaapeli kytketään USB-paikkaan tietokoneessa, mutta se on mahdollista myös kytkeä sarjaporttiin. Tämä riippuu käytettävästä kaapelista ja puhelimesta. Tietokoneeseen vaaditaan myös ohjelma, joka osaa kommunikoida matkapuhelimen ja PC:n välillä.

2.2.2 IrDA

IrDA-tiedonsiirtoyhteyttä käytettäessä puhelin kommunikoi toisen laitteen kanssa infrapunayhteydellä. Data siirretään muutamassa sekunnissa laitteesta toiseen tietenkin tiedoston koon mukaan. Siirrettäessä dataa laitteiden pitää olla näköetäisyydellä toisistaan, mikä tarkoittaa käytännössä noin 20 cm maksimissaan.

2.2.3 Bluetooth

Bluetoothilla pystytään siirtämään dataa ilman näköyhteyttä. Laitteiden välillä voi olla maksimissaan noin 10 m väliä, jotta ne vielä pystyvät kommunikoimaan. Etäisyys on tietenkin verrannollinen mahdollisiin esteisiin. Esimerkiksi seinät kommunikoivien laitteiden välillä vähentävät tiedonsiirron onnistumisen mahdollisuutta. Bluetooth-menetelmässä laitteet kommunikoivat toistensa kanssa radioaaltojen avulla.

2.2.4 MMC

Muistikorttien nopea kehitys on kasvattanut ensimmäisten 16 megatavujen muistikorttien kapasiteetin nykyään jo gigatavuun. Uusimpiin puhelimiin on tullut myös jo Mini SD -kortit, joita on nykyään jo kahden gigatavun kokoisia. MMC:llä eli muistikortilla voidaan siirtää tiedostoja laitteesta toiseen. MMC voidaan vaihtaa lennossa toiseen ilman puhelimen uudelleen käynnistämistä. Datan siirto puhe-

limen ja tietokoneen välillä on myös mahdollista, mutta silloin tarvitaan erillinen muistikortinlukija tietokoneeseen. Tietysti on myös mahdollista siirtää dataa infrapuna- tai Bluetooth-yhteydellä, mutta tällöin tietokoneeseen tarvitaan erilliset laitteet.

2.3 Näytön ominaisuudet

Series 60 -matkapuhelin luo uudenlaisen lähestymistavan matkapuhelimiin. Matkapuhelimien näytöistä on tehty melko suuria, jotta ne soveltuisivat hyvin erilaisten sovellusten tarpeisiin. Seuraavanlaisia näytön kokoja tuetaan tällä hetkellä: 176 x 208, 240 x 320 ja 352 x 416 pikseliä [3]. Näyttö voidaan vaihtaa pystyasennosta vaaka-asentoon esimerkiksi kesken videon katselun. Series 60 -matkapuhelimissa on värinäyttö, mutta värien lukumäärän ja näytön resoluution suunnittelee puhelimen valmistaja itse. Näytön koko, tarkkuus ja väriominaisuudet antavat entistä laajemmat mahdollisuudet erilaisten multimediasovellusten kehittämiselle.

2.4 Symbian-käyttöjärjestelmä

Series 60 -ohjelmistoalusta on luotu Symbian OS -käyttöjärjestelmän päälle ja se on siksi erittäin tärkeä osa-alue koko Series 60 -sarjan matkapuhelimille.

Symbian Ltd -yritys perustettiin vuonna 1998 Ericssonin, Nokian, Motorolan ja Psionin toimesta. Nykyään yrityksen omistajina ovat Ericsson, Panasonic, Samsung, Siemens, Sony Ericsson ja Nokia, joka omistaa lähes puolet Symbianista. [4]

Symbian OS -käyttöjärjestelmän käyttö perustuu lisenssiin, jonka yritys voi hankkia. Lisensoijia ovat tällä hetkellä maailman johtavat matkapuhelinvalmistajat: Arima, BenQ, Fujitsu, Lenovo, LG Electronics, Motorola, Nokia, Panasonic, Samsung, Sanyo, Sendo, Sharp, Siemens ja Sony Ericsson. [4]

Nokia käyttää Symbian OS -käyttöjärjestelmää esimerkiksi Series 60 -sarjan matkapuhelimissaan.

2.5 Symbian-käyttöjärjestelmän merkitys

Symbian OS on suunniteltu erityisesti langattomiin matkapuhelinlaitteisiin. Tarkoituksena on tarjota alusta, joka kuluttaa mahdollisimman vähän akku- ja muistikapasiteettia. Symbian on luonut luotettavan alustan uusille teknologioille, kuten GPRS:lle, Bluetoothille, SyncML:lle ja 3G-järjestelmälle. [4]

Kahdella fyysisesti ja käyttöliittymältään erilaisella matkapuhelimella voi olla samanlainen Symbian OS -koodi. Tämä koodi on yhteensopiva erilaisille matkapuhelimille. Tarkoituksena onkin yksinkertaistaa sovelluksien tekoa ja sovelluksien yhteensopivuutta eri laitteiden kanssa.

Symbian kehittää jatkuvasti Symbian OS -ohjelmistoa lisensoijien tarpeisiin. Symbian OS -käyttöjärjestelmän uusin versio on v9. Tämä käyttöjärjestelmä ei ole vielä käytössä yhdessäkään myynnissä olevassa matkapuhelimessa, mutta Nokian julkaisemaan N91 -puhelimeen se tulee.

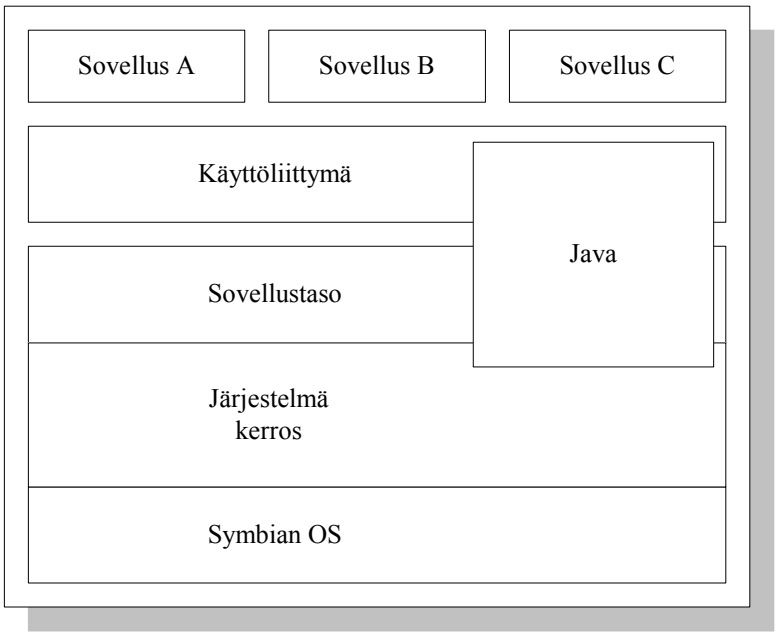
2.6 Arkkitehtuuri

Kuvassa 3 on kuvattuna Series 60 -ohjelmistoalustan arkkitehtuuri. Kuva antaa kokonaiskuvan käytetystä arkkitehtuurimallista.

Kernel eli niin sanottu käyttöjärjestelmän ydin sijaitsee Symbian OS -kerroksessa. Samassa kerroksessa ovat myös tiedostopalvelin, muistinhallinta ja laiteajurit. [2] Tämä kerros ohjaa sovelluksia ja hoitaa yhteyden käyttäjän ja laitteen välillä.

Järjestelmäkerros hoitaa viestipalveluja ja tietokantajärjestelyt. Esimerkiksi TCP/IP-, IMAP- ja SMS-protokollien käsittely hoidetaan tästä kerroksesta. [2]

Sovellustasokerros antaa ohjelmiston kehittäjälle mahdollisuuden vapaasti luoda omia sovelluksiaan käyttöjärjestelmään. Tämä on mahdollista, koska Series 60 on avoin ohjelmistoalusta. Ohjelmistojen kehittämiseen käytettäviä kieliä ovat Java ja C++. [2] Javasta on kehittynyt viime aikoina erittäin suosittu ohjelmointikieli langattomiin laitteisiin.



Kuva 3. Series 60 -ohjelmistoalustan arkkitehtuuri [2]

Käyttöliittymäkerros on luotu Symbian OS -käyttöjärjestelmän päälle. Tämä kerros huolehtii käyttäjän valitseman sovelluksen ja käyttöjärjestelmän välisestä kommunikoinnista. [2]

3 TESTAUKSEN MÄÄRITELMIÄ

Myersin mukaan testaus tarkoittaa ”ohjelman suorittamista virheiden löytämiseksi”. Myersin mielestä monet esittävät tästä prosessista virheellisiä määritelmiä, kuten esimerkiksi: ”Testauksella osoitetaan, että virheitä ei esiinny”, ”Testauksen tarkoitus on osoittaa, että ohjelma suorittaa sille määritellyt toiminnot oikein” ja

”Testauksella varmistutaan siitä, että ohjelma tekee sen, mitä sen oletetaan tekevän”. [5]

Testaus tapahtuu usein kokeilemalla ohjelmaa umpimähkäisesti jollain testiaineistolla. Ohjelmistojen testaus ei kuitenkaan ole mitään summittaista kokeilemista, vaan systemaattinen menetelmä virheiden eliminoimiseksi ohjelmistoista. Testausprosessissa testaus jaetaan useisiin vaiheisiin kuten testauksen kokonaisuuden suunnitteluun, testiympäristön määrittelyyn, testitapausten suunnitteluun, testien suorittamiseen sekä tulosten analysointiin ja raportointiin. [6],[7]

Tärkein syy testaamiseen on se, että kaikki ihmisten kirjoittamat ohjelmat sisältävät virheitä. Jo niin aikaisin, kuin vuonna 1949 Maurice Wilkes havaitsi seuraavaa: ”Se oli yksi minun matkoistani EDSAC-huoneen ja näppäilyvälineistön välillä, kun oivallus kulki koko voimallaan ylitseni ja tajusin, että suuri osa loppuelämästäni tulee kulumaan omien virheideni etsimiseen omista ohjelmistani”. [8]

Kuten Maurice Wilkes totesi, ohjelmistojen virheiden etsimiseen kuluu paljon aikaa. Myöhemmin on todettu, että ohjelmistotuotannossa testaukseen kuluva aika on usein jopa puolet koko ohjelman toteutukseen käytetystä ajasta [1].

Testaus on myös tärkeä osa koko prosessia, jolla pyritään varmistamaan toteutettavan tuotteen laatu. Vaikka laadunvarmistus kuuluu kaikkiin toteutusprosessin vaiheisiin, vain testaus on se vaihe, jonka avulla voidaan osoittaa, toimiiko ohjelma sen vaatimusten mukaan. Testauksen avulla voidaan saada myös paljon palautetta koko laadunvarmistukselle, jolloin prosessin aiemmissa vaiheissa tehdyistä virheistä voidaan oppia ja tuottaa myöhemmin parempia ohjelmistoja. [6],[7],[9]

3.1 Suunnitelmallisen testauksen periaatteet

Jotta testausta voidaan suunnitella ja suorittaa tehokkaasti, pitää ensin ymmärtää testauksen perusperiaatteet, jotka ohjaavat ohjelmistojen testausta: [7]

- Kaikkien testien pitää olla jäljitettävissä järjestelmälle asetettuihin vaatimuksiin.
- Kaikkien testien pitää olla toistettavissa.
- Testitapausten pitää olla suunniteltuina ennen kuin itse testaus alkaa.
- Pitää aloittaa ”pienestä” ja edetä kohti ”suurempaa” testausta.
- Täydellinen testaaminen on mahdotonta.
- Jotta testaus olisi mahdollisimman tehokasta, pitää jonkin suunnittelusta ja ohjelmoimisesta ulkopuolisen osapuolen osallistua testaukseen.
- Ohjelmoija ei saa olla ensisijainen testaaja.
- On käytettävä sekä lasi- että mustalaatikkotestausta.

Näiden periaatteiden mukaan suunnitellaan koko testaus. Testauksen suunnittelu muodostuu useista vaiheista kuten testaustasojen ja niiden sisään- ja ulospääsykriteerien määrittelystä, testausympäristöjen määrittelystä, testausmenetelmien ja välineiden valitsemisesta sekä testitapausten suunnittelusta. Testitapauksilla kuvataan, mikä on yksittäisen testin tavoite ja kuinka se suoritetaan. Testitapausten suunnittelu on testauksen suunnittelun tärkein osa-alue, koska niiden avulla suori-

tetaan itse testaus. Jos testitapaukset on suunniteltu huonosti, on testauksen laatu huonoa, eikä testauksella saavuteta sille asetettuja tavoitteita. [9]

3.2 Testauksen tavoitteet

Ohjelmistoja testataan seuraavien syiden takia: [9]

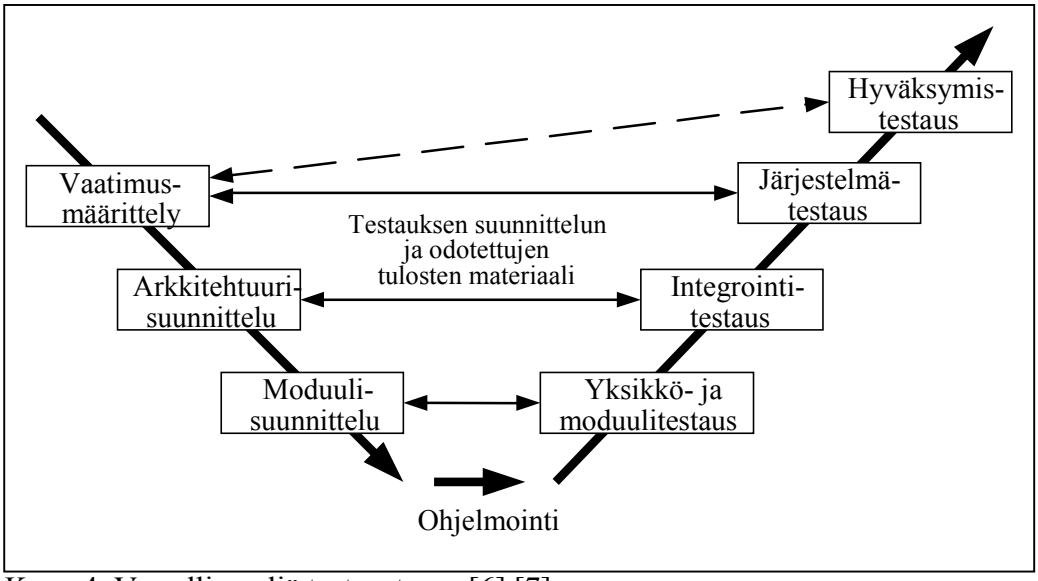
- Annetaan ohjelmoijille tietoa, jolla ehkäistään uusien virheiden syntymistä.
- Annetaan hallinnollista tietoa, jota tarvitaan ohjelmiston toteutuksen riskien evaluointiin.
- Päästään tilanteen mukaan mahdollisimman virheettömiin ohjelmistoihin.
- Luodaan testattavia malleja, joita voidaan helposti hyväksyä, hylätä ja ylläpitää.
- Voidaan hyväksyä ohjelmisto eli osoittaa, että se toimii.

Testauksen ensimmäinen tavoite on etsiä ohjelmistoista virheitä. Samalla testauksen tavoitteena on varmistaa, että asiakas on tyytyväinen saamaansa ohjelmistoon ja että ohjelmisto toimii asiakkaan vaatimusten mukaan. Virheiden etsimistä vielä tärkeämpi tavoite on virheiden ehkäiseminen. Torjuttu virhe on nimittäin paljon parempi kuin löydetty virhe, koska mitä aikaisemmin virhe löydetään, sitä halvempaa sen korjaaminen on. Torjuttu virhe ei esimerkiksi vaikuta projektin aikatauluihin ja resursseihin, koska sitä ei tarvitse testata uudelleen sen varmistamiseksi, että virhe on todellakin korjattu. Löydetylle virheelle taas pitää aina varata virheen korjaaja ja korjauksen testaaja. Lisäksi on aina vaarana, että korjaus aiheuttaa uusia virheitä ja lisää siten edelleen työmäärää.

Testaus ja virheenjäljitys liitetään yleensä toisiinsa ja niiden roolit sekoitetaan usein. Monelle nämä ovat toistensa synonyymeja, mutta asia ei ole näin. Testauksen tarkoitus on osoittaa, että ohjelmistossa on vikoja. Viat ovat määrittelystä poikkeavaa toimintaa, kun taas virheet ovat koodissa, suunnittelussa tai määrittelyssä olevia todellisia virheitä. Ja virheenjäljityksen tarkoitus on selvittää, missä vian aiheuttava varsinainen virhe sijaitsee sekä auttaa suunnittelemaan ja toteuttamaan muutokset, joilla vika saadaan korjattua. [6]

3.3 Testauksen tasot

Toteutusvaiheen jälkeinen testaus jaetaan yleensä kolmeen päätasoon: yksikkö- ja moduulitestaukseen, integrointitestaukseen ja järjestelmätestaukseen [6]. Jako testaustasoihin riippuu siitä, missä ympäristössä testaus suoritetaan, kuka sen suorittaa ja mistä lähteestä saadaan testauksessa käytetty materiaali. Kuvassa 4 esitellään testaustasojen selkein jakomalli eli V-malli. Kuvan 4 V-mallissa on kolmen päätason lisäksi määriteltynä vielä neljäs testauksen taso eli hyväksymistestaus.



Kuva 4. V-mallin neljä testaustasoa [6],[7]

Jokaiselle tasolle on määritelty sisään- ja ulospääsykriteerit. Sisäänpääsykriteerillä määritellään, milloin ohjelma tai moduuli on riittävän valmis, jotta se voidaan päästää tähän testauksen vaiheeseen. Esimerkiksi yksikkö voidaan ottaa yksikkötestaukseen vasta, kun se kääntyy ilman virheilmoituksia. Ulospääsykriteereillä valvotaan, että ohjelma tai moduuli on testattu riittävän hyvin kyseisellä tasolla ennen siirtymistä seuraavalle tasolle. Ulospääsykriteereinä voidaan käyttää esimerkiksi testauksen kattavuuden mittausta tai löydettyjen virheiden määrää. [6]

Mitä korkeammalla V-mallin testaustasolla virheitä löydetään, sitä kalliimmaksi niiden korjaaminen tulee. Jos virhe havaitaan vasta järjestelmätestauksessa, se voi aiheuttaa muutoksia useisiin moduuleihin. Jotta voidaan varmistua, että kaikki moduulit toimivat muutosten jälkeen oikein, ne joudutaan kaikki testaamaan uudelleen. Tätä uudelleen testausta kutsutaan regressiotestaukseksi [7]. Koska regressiotestauksessa toistetaan olemassa olevia testejä, se on eräs parhaiten automatisointiin soveltuva testauksen alue [9].

Monet nykyiset ohjelmistot ovat niin laajoja, että ne muodostuvat useista erillisistä ohjelmista. Kun tällaista ohjelmistoa testataan, jo sen moduulitestaustaso voi sisältää kaikki kolme päätestaustasoa jokaisen ohjelman osalta. Vasta kun kyseinen ohjelma on erikseen järjestelmätestattu, se voidaan hyväksyä mukaan koko ohjelmiston integrointitestaukseen. Kun eri ohjelmat on saatu integroitua yhdeksi ohjelmistoksi, ohjelmisto järjestelmätestataan vielä kokonaisuutena.

3.3.1 Yksikkö- ja moduulitestaus

Joissain esityksissä yksikkö- ja moduulitestaus erotellaan omiksi tasoikseen, mutta niitä voidaan hyvin pitää yhtenä tasona, koska molempien testit suoritetaan samassa ympäristössä ja koska molempien suunnittelun lähteenä toimii moduulisuunnittelu. Yksikkö- ja moduulitestaukseen käytetään yleensä lasilaatikkomenetelmää, jonka tavoitteena on käydä läpi yksikön tai moduulin kaikki loogiset rakenteet. Lasilaatikkomenetelmässä testattavan komponentin toiminta ja toteutus koodissa tunnetaan. [7],[9]

Yksikkötestauksessa testataan pienintä mahdollista ohjelman osaa, jota ylipäänsä pystytään testaamaan. Tällainen yksikkö on yleensä pienin ohjelman osa, joka pystytään kääntämään ja suorittamaan, esimerkiksi jokin aliohjelma. Yksikkötestauksen avulla varmistetaan, että kyseinen yksikkö toimii vaatimustensa ja suunnitelmiensa mukaan oikein. Siinä testataan vain yksikön sisäinen toiminta ja esimerkiksi kutsuja muihin aliohjelmiin ei testata. Moduulitestauksessa testataan useista yksiköistä koottujen isompien kokonaisuuksien eli moduulien toimivuutta. Tavoitteena on varmistaa koko moduulin toimivuus testaamalla yksikköjen välisiä rajapintoja. Yleensä tätä varten joudutaan tekemään pieniä apu- ja tynkäohjelmia, jotta pystytään testaamaan kaikkia toimintoja. Yksikkö- ja moduulitestauksien etuna on, että mahdolliset virheet pystytään rajaamaan erittäin pienelle alueelle, jolloin niiden korjaaminen on helpompaa. Yksikkö- ja moduulitestauksen suorittaa yleensä aina moduulin tekijä itse. [1],[6],[9]

3.3.2 Integrointitestaus

Integrointitestauksessa testataan moduuleista yhteen koottua kokonaisuutta. Sen avulla varmistetaan, että kaikki yksinään testatut moduulit toimivat myös yhdessä. Integrointitestauksessa keskitytään moduulien välisen yhteistyön ja niiden rajapintojen testaamiseen. Lisäksi integrointitestauksessa varmistetaan, ettei moduuleilla ole sivuvaikutuksia. Integrointitestauksen lähteenä toimii arkkitehtuurisuunnitelma. Yleensä erilliset integrointitestaajat vastaavat integrointitestauksen suunnittelusta ja suorittamisesta. Integrointitestaus etenee integroinnin mukana joko kokoavasti (bottom up) alimman tason moduuleista ylöspäin tai päinvastaisesta etenemissuunnasta (top down). Moduulien kokoamisjärjestys vastaa niiden välisiä kutsusuhteita. Myös niin sanottu "big bang" -menetelmä, jossa kaikki kootaan kerralla yhdeksi kokonaisuudeksi, on usein käytetty. Tämän menetelmän ongelmana on virheen aiheuttavan moduulin hankala paikantaminen, koska kaikki moduulit ovat testauksessa jo mukana. [6],[7],[9]

3.3.3 Järjestelmä- ja hyväksymistestaus

Järjestelmätestauksessa testataan koko järjestelmä todellista käyttöympäristöä vastaavassa ympäristössä. Tämä on koko testauksen kaikkein vaativin vaihe. Vain sen avulla voidaan varmistaa, että koko järjestelmä toimii vaatimustensa mukaan. Järjestelmätestaukseen käytetään pääasiassa mustalaatikkomenetelmää. Mustalaatikkomenetelmässä ei tiedetä testattavan systeemin toteutuksesta eikä sen sisäisestä toiminnasta mitään. On erittäin tärkeää, että ohjelmiston kehitystyöstä riippumattomat testaajat suunnittelevat ja suorittavat järjestelmätestauksen. [1],[5],[9] Järjestelmätestaus voidaan jakaa osa-alueisiin. 1970-luvun loppupuolella esiteltiin seuraavat 15 osa-aluetta: [5]

- dokumentaatiotestaus
- huollettavuustestaus
- käytettävyystestaus
- käyttöönotto- ja asennustestaus
- käyttöproseduuritestaus
- luotettavuustestaus
- massatestaus
- rasitustestaus
- regressiotestaus

- suorituskykytestaus
- tallennus ja muistin testaus
- tietoturvallisuustestaus
- toiminnallisuustestaus
- toipumistestaus
- yhteensopivuustestaus käyttöympäristön kanssa.

Seuraavissa alakohdissa esitellään näistä vaiheista tämän työn kannalta tärkeimmät, joita ovat regressiotestaus, rasitustestaus ja toiminnallisuustestaus.

3.3.3.1 Regressiotestaus

Ohjelmiston kehitysvaiheessa uuden koodin lisääminen ja muutoksien tekeminen saattavat vaikuttaa ohjelmiston eri osien toimintaan. Nämä muutokset saattavat aiheuttaa sen, että ennen toimineet osat eivät toimi enää muutoksien jälkeen. Tämän vuoksi käytetään regressiotestausta varmistamaan, että ohjelmisto toimii oikein ja vaatimusten mukaisesti muutoksien jälkeen. Regressiotestauksen päätavoitteena on havaita ohjelmiston kehitys- ja ylläpitovaiheessa syntyneet virheet. [7],[10]

3.3.3.2 Rasitustestaus

Rasitustestauksessa ohjelmistoa tai sen komponenttia yritetään kuormittaa äärimmäisyyksiin. Tämän seurauksena ohjelmistosta pyritään löytämään heikot ja vialliset kohdat. Jos vikakohta löydetään, odotettu tulos on hallittu virhetilanne. Hyvänä esimerkkinä on Internet-sivuston rasitustestaus, jossa käyttäjämäärä saattaa olla suuri ja halutaan varmistaa, että sivut kestävät kuormaa sallituissa rajoissa ja eivät aiheuta pysyviä vahinkoja virhetilanteissa. Rasitustestaus on yleensä hyödyllistä silloin, kun testataan isoja järjestelmiä. [10]

3.3.3.3 Toiminnallisuustestaus

Toiminnallisuustestauksen tulisi testata kaikki järjestelmän ominaisuudet. Se seuraa yleensä yksikkö- ja integrointitestauksen vaiheita. Toiminnallisuustestaus keskittyy ohjelman vaatimusten testaamiseen ja sen tavoitteena on löytää eroja ohjelman määrittelyn ja toteutuksen välille. Toiminnallisuustestaus on käytännössä mustalaatikkotestausta järjestelmätestausvaiheessa. [10]

3.3.4 Hyväksymistestaus

Hyväksymistestausta pidetään joskus osana järjestelmätestausta. Koska se suoritetaan yleensä eri ympäristössä ja eri henkilöiden toimesta kuin järjestelmätestaus, se sopii paremmin omaksi tasoksi. Yleensä hyväksymistestauksessa koko järjestelmä testataan vielä kertaalleen ennen käyttöönottoa asiakkaan omassa ympäristössä, jolloin se suoritetaan yleensä yhteistyössä asiakkaan kanssa. Hyväksymistestauksessa varmistetaan, että koko ohjelmisto ja sen käytettävyys on asiakkaan odotusten mukainen. Samalla varmistetaan, ettei asiakkaan ympäristössä ole mitään tuntemattomia komponentteja, joiden sivuvaikutukset haittaisivat järjestelmän toimintaa. [1]

3.4 Testitapausten suunnittelu

Testitapauksia suunniteltaessa on tavoitteena laatia mahdollisimman tehokkaita ja hyödyllisiä testitapauksia, jotta niitä suorittamalla koko testaus olisi laadukasta. Hyvän testitapauksen tunnusmerkkejä ovat: [7]

- Testitapauksella on suuri todennäköisyys löytää virheitä.
- Testitapaus ei ole tarpeeton. Koska aika ja resurssit ovat rajoitettuja, turhien ja samaa asiaa testaavien testitapausten suorittamisessa ei ole järkeä.
- Testitapauksen pitää olla paras omassa luokassaan. Jos on olemassa samantyyppisiä testitapauksia, niistä pitää valita se, joka kattaa parhaiten koko luokan tavoitteet.
- Testitapaus ei saa olla liian yksinkertainen eikä liian monimutkainen.

Jotta testaaja pystyy suunnittelemaan tällaisia testitapauksia, pitää hänellä olla joko erinomainen kuva koko ohjelmiston toiminnasta tai erittäin tarkat vaatimusmäärittelyt ja toteutussuunnitelmat. Testitapausten suunnitteluun on olemassa useita menetelmiä, mutta yleensä testitapaukset valitaan kahdesta lähestymistavasta eli joko lasilaatikkotestauksesta tai mustalaatikkotestauksesta. Näitä kahta menetelmää yhdistelemällä saadaan harmaalaatikkotestaus, joka hyödyntää molempien menetelmien parhaita puolia. [5],[7],[9]

4 TESTAUKSEN AUTOMATISOINTI

Ohjelmistojen testauksen pitää olla kattavaa, jotta se löytäisi mahdollisimman suuren osan ohjelmistossa olevista virheistä. Kattavuuden lisäksi testauksen pitää olla myös mahdollisimman suorituskykyistä, eli testaus pitää kyetä suorittamaan mahdollisimman nopeasti ja halvalla. Testauksen automatisoinnilla voidaan nopeuttaa ja parantaa toimivaa testausta. Automatisoimalla testaus voidaan lyhyessä ajassa suorittaa paljon enemmän testejä kuin manuaalisesti. Testejä, jotka manuaalisesti suoritettuna kestävät useita tunteja, voidaan usein suorittaa minuuteissa tai niiden suoritus voidaan siirtää yöaikaan tapahtuvaksi. Ennen kaikkea regressiotestausta voidaan suorittaa useammin, nopeammin ja tarkemmin automatisoitujen testien avulla. Sen avulla on saavutettu jopa 80 % säästöjä manuaaliseen testaukseen verrattuna [11]. Joissain yrityksissä on testauksen automatisoinnin avulla pystytty säästöjen sijaan tuottamaan parempilaatuisia ohjelmistoja nopeammin kuin käyttäen manuaalista testausta.

Automatisoinnilla ei voida korvata manuaalista testausta eikä testauksen suunnittelua [11]. Testauksen suunnittelua ja manuaalista testausta voidaan kuitenkin parantaa automatisoinnin avulla. Esimerkiksi testauksen suunnitteluun ja manuaalisten testien suorittamiseen jää enemmän aikaa, jos suuri osa regressiotestausta on automatisoitu. Testauksen ja automatisoinnin suunnittelussa on tärkeä muistaa, ettei sillä, suoritetaanko testi automaattisesti vai manuaalisesti, ole mitään vaikutusta siihen, kuinka tehokas ja hyvä itse testi on. Jos testeillä ei saavuteta mitään tuloksia manuaalisesti, ei niiden suorittamisesta nopeammin ole paljoakaan hyötyä.

Testauksen automatisointi on kuitenkin hyvin vaikeaa. Monet organisaatiot yllättyvätkin automatisoinnin vaatimasta suuresta työmäärästä. Jo yhden testin auto-

matisointi vie yleensä huomattavasti enemmän aikaa kuin sen suorittaminen manuaalisesti kertaalleen. Suurin yllätys aloitteleville automatisoijille on kuitenkin työn luonne: testauksen automatisointi on ohjelmointia. Siksi se vaatii hyvin erilaisia taitoja kuin testaaminen, ja usein hyvilläkään testaajilla ei ole riittäviä taitoja suunnitella ja toteuttaa sitä tarpeeksi hyvin. Kunnollinen ja tehokas testauksen automatisointi vaatiikin siihen erikoistuneita ammattilaisia. [9],[11],[12],[13]

Testauksen automatisointina pidetään yleensä testitapausten suorittamista automaattisesti erilaisten työkalujen avulla. Työkalu hoitaa syötteiden antamisen testaajan puolesta eli se simuloi käyttäjää testissä. Usein työkalu hoitaa myös tulosten vertaamisen ja niiden keräämisen jälkikäteen suoritettavaa analysointia varten. Testauksen automatisointi voi sisältää kuitenkin paljon enemmän ja sitä voidaan hyödyntää lähes kaikissa testauksen vaiheissa. Testaustyökalujen tarjoamia mahdollisuuksia esitellään tarkemmin luvussa 4.2. Kokonaisuutena testauksen automatisointi tarkoittaa toimivan manuaalisen testausprosessin automatisointia. Jotta tämä olisi mahdollista, pitää olla olemassa täsmällisesti määritelty manuaalinen testausprosessi. [9],[13] Ilman sitä automatisoinnilla ei voida saavuttaa todellista hyötyä. Testauksen suorituksen automatisoiminen on kuitenkin ehdottomasti tärkein ja nopeimmin kasvava testauksen automatisoinnin osa-alue.

Jotta testitapauksia pystyttäisiin suorittamaan automaattisesti, pitää niitä varten olla suoritettava testauskoodi [11]. Yksinkertaisimmillaan testauskoodi sisältää vain syötteen ja siitä saatavan tuloksen kirjauksen. Tällaisten testauskoodien avulla voidaan automatisoida merkkipohjaisten käyttöliittymien testausta.

Graafisten käyttöliittymien testauksen automatisointi vaatii paljon enemmän työtä, koska niissä on jo syötteiden antamiseen useita tapoja. Helpoin tapa graafisten käyttöliittymien vaatimien testauskoodien tekemiseen on manuaalisesti suoritettavan testitapauksen aikaansaaman tapahtumien ketjun nauhoittaminen. Tätä menetelmää kutsutaan nauhoita ja toista -menetelmäksi. Työkalu hoitaa automatisoinnin nauhoittamalla testauskoodin kielellä testaajan hiiren liikkeet, kenttien syötteet, valikkojen valinnat jne. Tämän jälkeen nauhoituksen tapahtumat voidaan toistaa automaattisesti. Todellisuudessa automatisointi ei ole näin helppoa, sillä esimerkiksi jonkin valikon paikka voi olla seuraavalla kerralla vaihtunut ja silloin testi ei enää toimikaan. Kunnollinen testauksen automatisointi vaatii yleensä 3–10 kertaa enemmän aikaa kuin testitapausten suorittaminen kertaalleen manuaalisesti. [13]

4.1 Testauksen automatisoinnilla saavutettavat hyödyt

Automatisoidulla testauksella voidaan saavuttaa merkittäviä hyötyjä verrattuna manuaaliseen testaukseen, kun se suunnitellaan ja toteutetaan huolellisesti. Sillä voidaan saavuttaa kolme tärkeää hyötyä: luotettavampien järjestelmien tuottaminen, testaustyön laadun paraneminen sekä testaukseen käytetyn työmäärän väheneminen. Tarkemmin jaoteltuna automatisoinnilla voidaan saavuttaa seuraavia hyötyjä verrattuna manuaaliseen testaukseen: [11],[12]

- Saadaan paljon tehokkaampi isku- ja regressiotestaus.
- Testit voidaan suorittaa nopeammin ja useammin.
- Testaus muuttuu mielenkiintoisemmaksi.
- Resurssit saadaan paremmin käyttöön.

- Testitapausten uudelleenkäyttö kasvaa.
- Tuotetaan johdonmukaisempia ja toistettavampia testitapauksia.
- Vähennetään inhimillisiä virheitä.
- Saavutetaan varmempia tuloksia.
- Suorituskykytestit on helpompi suorittaa.

Iskutestauksella varmistetaan nopeasti, että ohjelmalle kannattaa aloittaa varsinainen testaus. Iskutestauksella testataan ohjelmiston tärkeimmät ominaisuudet, jotka vaikuttavat muuhun testaukseen. Iskutestaus ja regressiotestaus ovat paljon helpompia suorittaa ohjelman uudelle versiolle, jos automatisoidut testit ovat valmiina edelliseltä kierrokselta. Nämä myös kannattaa ensimmäiseksi automatisoida. Myös kynnys suorittaa testejä madaltuu, jos ne ovat helpommin ja nopeammin suoritettavissa. Siksi testaajat usein huomaavatkin, että he suorittavat automatisoituja testejä paljon useammin kuin vastaavia manuaalisia. Testaajat voivat keskittyä mielenkiintoisempiin töihin, koska tylsä ja yksinkertainen työ, kuten samojen syötteiden toistaminen uudelleen ja uudelleen, siirtyy työkalujen tehtäväksi. [9] Koneet saadaan automatisoinnin avulla myös yöksi hyötykäyttöön ja näin testaajat voivat keskittyä uusien testitapausten suunnittelemiseen ja manuaalisten testien suorittamiseen. Näin saadaan kokonaisuutena kaikki resurssit paljon parempaan käyttöön. Myös testitapausten uudelleenkäyttö tehostaa resurssien käyttöä, koska valmiita automatisoituja testejä voidaan hyödyntää kaikilla testaustasoilla helposti. Samalla näistä testitapauksista tulee laadultaan paljon parempia, koska niillä on useita käyttäjiä ja niiden suunnitteluun voidaan uhrata enemmän aikaa.

Automatisoidut testitapaukset toistetaan aina täsmälleen samoin, mikä vähentää merkittävästi testin tuloksiin vaikuttavien inhimillisten virheiden määrää. Löydetyt virheet on tällöin myös huomattavasti helpompi jäljittää. Testien hajauttaminen on paljon helpompaa ja nopeampaa kuin manuaalisesti. Jos esimerkiksi joudutaan testaamaan manuaalisesti neljällä eri alustalla, se vie aikaa neljä kertaa niin paljon kuin testattaessa vain yhdellä alustalla. Automatisoinnilla on mahdollista saavuttaa tilanne, jossa on aivan sama, testataanko yhdellä vai kymmenellä alustalla, jolloin saadaan paljon nopeammin paljon kattavampia tuloksia. Kun tällainen laaja automatisoitu testimateriaali on saatu suoritettua onnistuneesti, ollaan paljon varmempia siitä, ettei ikäviä yllätyksiä ilmaannu, kun ohjelmisto julkaistaan. Manuaalisesti on myös lähes mahdotonta simuloida 1000 yhtäaikaista käyttäjää, mutta työkalujen avulla se voi olla yhtä helppoa kuin viiden käyttäjän simuloiminen. Siksi jonkinasteinen automatisointi onkin lähes välttämätöntä suorituskykytestauksessa. [9]

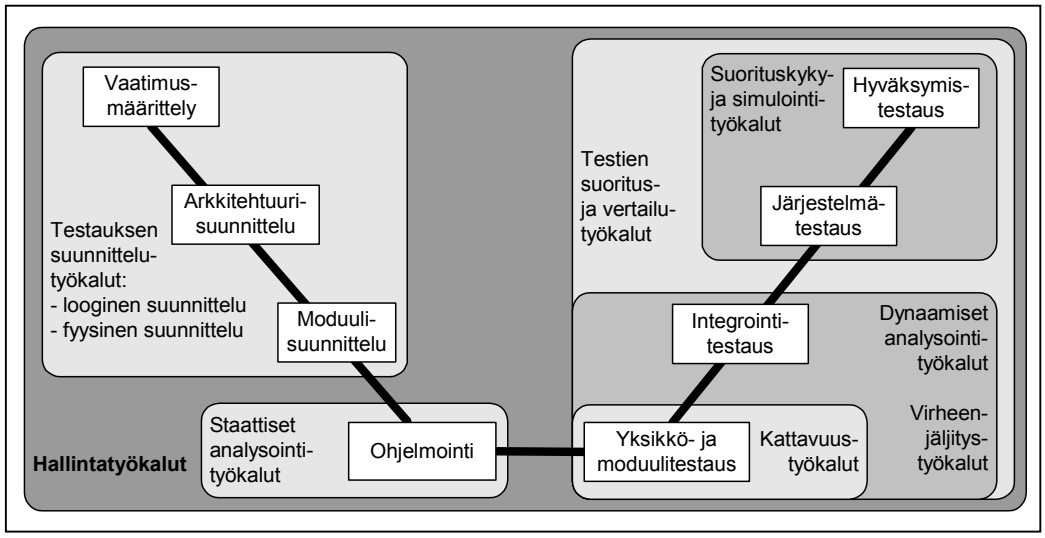
Pitää kuitenkin ymmärtää, että näiden hyötyjen saavuttaminen vaatii oman merkittävän työpanoksensa ja niihin liittyy suuri määrä omia ongelmia, jotka pitää saada ratkaistua. Automatisointiin liittyviä yleisimpiä ongelmia käsitelläänkin tarkemmin luvussa 4.3.

4.2 Testauksen automatisoinnin tuki eri ohjelmistonkehityksen vaiheisiin

Testauksen automatisointi voi olla paljon muutakin kuin pelkästään testitapausten automaattista suorittamista ja tulosten keräämistä. Työkaluja on saatavilla testauksen avuksi kaikkiin ohjelmistonkehityksen vaiheisiin. Niitä voidaan hyödyntää jo niinkin aikaisessa vaiheessa kuin vaatimusten määrittelyssä. [9] On olemassa esimerkiksi työkaluja, jotka käyvät läpi vaatimuksia ja analysoivat niiden tekstiä. Jos

tekstin tyyli tai käytetyt sanamuodot eri vaatimuksissa ovat hyvin erilaisia, voidaan epäillä, etteivät vaatimukset ole tarpeeksi selkeästi määriteltyjä. Testaajan on vaikea havaita tällaisia puutteita vaatimuksissa, mutta työkalu käy ne kaikki samalla tavalla läpi ja auttaa siten testaajaa havaitsemaan poikkeamat. Kuvassa 5 on esitelty, mihin kaikkiin ohjelmistonkehityksen vaiheisiin on saatavilla testaustyökaluja.

Hallintatyökaluilla tarkoitetaan työkaluja, joita voidaan käyttää apuna testauksen suunnittelussa, testitapausten ja vaatimusten välisten riippuvuuksien hallinnassa, jo suoritettujen testitapausten sekä löydettyjen virheiden käsittelyn seuraamisessa. Esimerkiksi vaatimusten ja testitapausten riippuvuuksien hallinta, ja sitä kautta testauksen kattavuus suhteessa vaatimuksiin, on huomattavasti helpompi hoitaa kunnollisten työkalujen avulla verrattuna manuaaliseen dokumenttien päivittämiseen. Jos projektin jossain vaiheessa on niin kiire, ettei kaikkea dokumentaatiota ehditä päivittää, voidaan jotkin asiat unohtaa kokonaan. Jos käytössä on kunnolliset työkalut, päivitys tulee kerralla koko dokumentaatioon, eikä mitään tietoa kadoteta. Testitapausgeneraattorit ovat loogisen suunnittelun työkaluja. Niillä yritetään luoda testitapauksia vaatimusten, käyttöliittymien tai koodin logiikan avulla. Fyysisen suunnittelun työkaluilla taas käsitellään tai luodaan uutta testimateriaalia. Esimerkiksi työkalu, joka poimii satunnaisia tietueita tietokannasta, on fyysisen suunnittelun työkalu. Staattiset analysointityökalut käyvät koodia läpi suorittamatta sitä. Ne löytävät koodista tietyntyyppisiä virheitä, kuten alustamattomien muuttujien käyttö, paremmin kuin mikään muu menetelmä. [11]



Kuva 5. Mihin testaustyökalut sopivat ohjelmistonkehityksessä [11]

Kattavuustyökaluilla voidaan mitata, kuinka kattavasti ohjelma käydään läpi tietyillä testeillä. Niitä voidaan hyödyntää erityisesti yksikkö- ja moduulitestauksessa esimerkiksi silloin, kun halutaan varmistaa, että koodin haarakattavuus tai lausekattavuus on riittävä. Vaikka virheenjäljitystyökalut eivät ole varsinaisesti testaustyökaluja, niistä on silti erittäin paljon hyötyä, kun halutaan löytää jonkin virheen syy. [9] Dynaamisilla analysointityökaluilla seurataan järjestelmän tilaa ohjelmiston suoritusaikana etsien esimerkiksi muistivuotoja. Simulaattoreilla simuloidaan tilanteita, joita todellisuudessa ei voida kokeilla, kuten esimerkiksi ydinvoimalaonnettomuuksia. Suorituskykytestaus jakaantuu useampiin lajeihin kuten kuormitus-, rasitus- ja konfiguraatiotestaukseen, ja siksi siihen on olemassa paljon

erilaisia työkaluja. Niillä voidaan esimerkiksi mitata tiettyjen tapahtumien vaatimia aikoja tai toistaa samoja tapahtumia useita päiviä etsien mm. lukkiintumistilanteita. Kuormitustestaustyökaluilla voidaan luoda palvelimelle simuloituja yhtäaikaisten käyttäjien ja tapahtumien kuormia, joiden avulla voidaan selvittää mm. järjestelmän pullonkauloja ja maksimaalisia yhtäaikaisten käyttäjien määriä.

Testauksen suoritus- ja tulosten vertailutyökalut ovat kuitenkin suosituimmat testauksen apuvälineet. Niitä voidaan hyödyntää kaikilla testaustasoilla ja niihin kuuluvat mm. kaikki nauhoita ja toista -menetelmän työkalut. Suoritus- ja vertailutyökaluilla saavutetaan kaikkein eniten hyötyä erityisesti silloin, kun ollaan aloittamassa testauksen automatisointia. [11]

4.3 Testauksen automatisoinnin ongelmat

Testauksen automatisoinnilla saavutettavat potentiaaliset hyödyt ovat kiistattomat. Silti useat testauksen automatisointiprojektit ovat epäonnistuneet ja on palattu takaisin kokonaan manuaaliseen testaukseen. On useita ongelmia, joihin voidaan törmätä, kun automatisoidaan testausta. Jos niistä tiedetään jo etukäteen, niiltä on helpompi välttyä. Yleensä automatisoidessa törmätään helposti seuraavanlaisiin ongelmiin: [9],[11],[12]

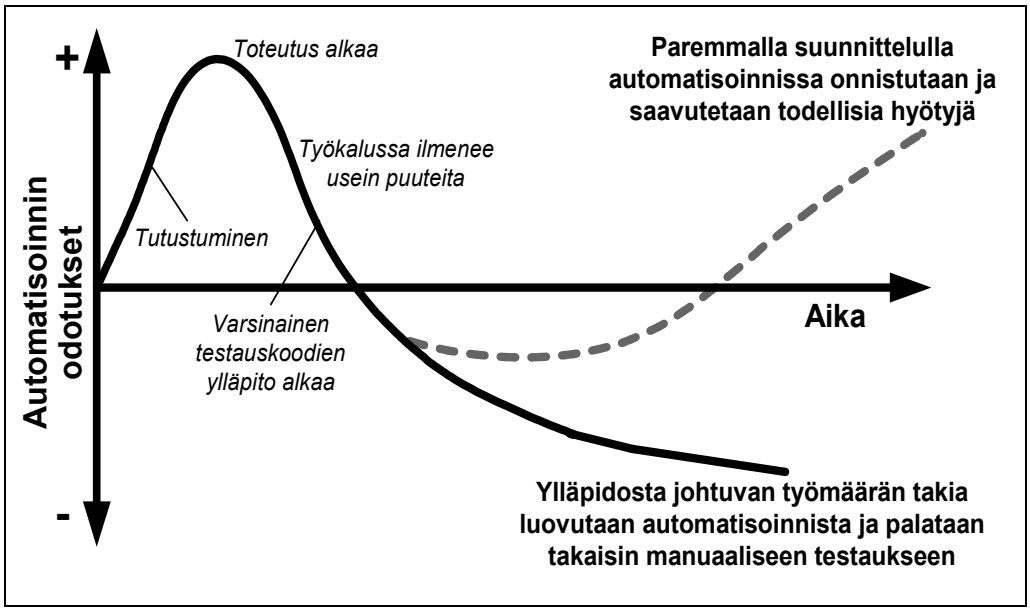
- epärealistiset odotukset saavutettavista hyödyistä
- tekniset ongelmat, kuten vaikeasti käytettävät työkalut
- organisaation ongelmat, kuten resursointi ja aikatauluttaminen
- huonot testausmenetelmät
  ”Kaaoksen automatisointi antaa vain nopeamman kaaoksen” [11]
- odotukset, että automatisoidut testit löytävät paljon uusia virheitä
- virheellinen varmuuden tuntemus
- testitapausten ylläpito
- nauhoita ja toista -menetelmän antama harhakuva automatisoinnin helppoudesta
- automatisoinnin vaatimien ohjelmointitaitojen puute
- testitapausten automaattinen generoiminen ilman automatisoitua suoritusympäristöä
- ainoastaan ennalta määriteltyjen toimintoketjujen testaaminen jättäen kokonaan testaamatta uusien käyttäjien usein epäloogiset toimintaketjut.

Usein ollaan erittäin optimistisia työkalujen suhteen ja oletetaan, että uusi teknologia ratkaisee kaikki ongelmat. Kaupalliset testaustyökalut ovat kuitenkin suuria ja monimutkaisia tuotteita, joiden käyttö vaatii opiskelua. Niiden tehokas hyödyntäminen voi olla vaikeaa ja nekin voivat sisältää virheitä. Pettymys voikin olla suuri, kun huomataan, että edes testaukseen tarkoitettua työkalua itseään ei ole testattu kunnolla. Tällaistakin voi valitettavasti sattua. Myös kaupallisten työkalujen valmistajat luovat epärealistisia odotuksia esittelemällä työkalujaan ratkaisuina, joiden avulla voi hoitaa kaiken testaukseen liittyvän hyvin pienellä työmäärällä.[12] Myös testattava ohjelma voi olla suunniteltu niin, että sen testaus on erittäin hankalaa manuaalisestikin. Tällaisen ohjelman testaaminen työkalujen avulla saattaa olla vielä paljon vaikeampaa.

Niinpä olisi erittäin tärkeää, että projektien vetäjät ja organisaatioiden johtajat perehtyisivät tarkemmin testauksen automatisointiin, ennen kuin asettavat sille ta-

voitteita. Jos johdon odotukset ovat liian korkeat, ei ole mitään merkitystä, kuinka hyvin työkalu tai automatisointi toimii, sillä se ei koskaan saavuta sille asetettuja odotuksia. Usein ongelmana on myös se, että automatisointia yritetään toteuttaa sivutoimena muun työn ohessa. Tällöin organisaatio ei varaa automatisointiin kunnollisia resursseja eikä aikaa, joten siihen perehtyminen ja sen suunnitteleminen jäävät puutteellisiksi. Puutteellinen resursointi yhdessä epärealististen odotusten kanssa aiheuttaa todella helposti kuvan 6 yhtenäisen mustan käyrän mukaisen käyttäytymisen ja automatisoinnista luopumisen.

Kaikki uuden oppiminen viehättää yleensä suunnattomasti. Kuitenkin jos testaus on laadullisesti heikkoa, ei automatisoinnissa ole mitään järkeä. Jos testit eivät löydä manuaalisesti virheitä, eivät ne löydä niitä automaattisestikaan suoritettuina. Sen ymmärtäminen ja myöntäminen, että oma työ on laadultaan heikkoa, voi olla erittäin hankalaa ja vastenmielistä. Silti on paljon järkevämpää parantaa ensin testauksen laatua kuin tehostaa huonolaatuista testausta.



Kuva 6. Automatisoinnin odotuksien käyrä []

Usein luullaan, että automatisoidut testit löytävät paljon virheitä. Todellisuudessa suurin osa virheistä löytyy, kun testi suoritetaan ensimmäistä kertaa, eli kun testi suoritetaan manuaalisesti. Vaikka testit eivät löytäisikään paljoa virheitä, on testien toistamisesta silti paljon hyötyä. Niillä saadaan helposti varmistettua, ettei jokin jo toimiva ominaisuus ole muuttunut uudessa versiossa, mikä lisää varmuutta tuotteen laatuun. Pitää kuitenkin varoa liiallista varmuuden tuntemusta. Se, että automatisoidut testitapaukset saadaan suoritettua onnistuneesti, ei tarkoita sitä, ettei ohjelmistossa olisi vielä virheitä. Jos esimerkiksi odotetut tulokset on määritelty väärin, työkalu antaa aina oikeat vastaukset, vaikka todellinen tulos olisi väärä.

5 TEST-X-TYÖKALU JA TESTAUS

Test-X on Tieto-X:n suunnittelema ja tekemä Series 60 -matkapuhelimien testaukseen kehitetty ja jatkuvasti kehittyvä testaustyökalu. Test-X:llä pystytään auto-

matisoimaan testausta ja muutenkin helpottamaan ja nopeuttamaan testauksen suorittamista. Se soveltuu siis matkapuhelimien järjestelmätestaukseen, koska se vaatii UI:n eli käyttöliittymän. Itse Test-X on työkalun ydin ja varsinaiset testit ovat plugineja eli lisäosia. Työkalu asennetaan joko puhelimen muistiin tai MMC:lle ja samalla lailla asennetaan erikseen testiapplikaatiot. Kuvassa 7 on testauksessa käytetyt matkapuhelimet eli vasemmalla on Nokian 6680 ja oikealla Nokian 6630. Nämä ovat siis molemmat Series 60 -ohjelmistopohjaisia matkapuhelimia.

Kuva 7. Nokian 6680 vasemmalla ja oikealla Nokian 6630

Markkinoilta olisi kyllä löytynyt matkapuhelimien järjestelmätestaukseen tehtyjä automatisointiohjelmia, mutta kuten luvussa 4.3 kerrottiin, niiden toimivuudesta ei koskaan tiedä. Tämän vuoksi päätettiin tehdä oma työkalu, joka on juuri sellainen kuin halutaan.

5.1 Test-X:n ominaisuudet

Test-X ja testit ovat SIS-paketteja, jotka asennetaan puhelimeen. Kuvassa 8 on näkymä käyttöjärjestelmästä, kun Test-X on siihen asennettu. Sille tulee oma ikoni, josta se käynnistetään.

Kuvassa 9 puolestaan on näkymä, kun Test-X on käynnistetty. Tämä on Test-X:n päänäkymä. Kuvasta nähdään, että siinä ovat Bluetooth (BT Tester), muistinhallinta (Mem Manager) ja asetusten tarkistus (Settings Check) testiapplikaatiot asennettuina. Tästä näkymästä voidaan valita, mitä tehdään.

Kuva 8. Test-X:n ikoni

Kuva 9. Test-X:n perusnäkymä

Vaihtoehdot ovat seuraavat: valitaan, mitä testiä ryhdytään suorittamaan, luodaanko vai muokataanko testisettiä vai laitetaanko valmiina oleva testisetti ajoon. Lisäksi voidaan katsoa lokitiedostoja, testiapplikaatioiden versionumeroita tai ohjeista apua ongelmaan. Viimeisenä vaihtoehtona voidaan tietysti valita Test-X:n sulkeminen. Seuraavaksi esitetään seuraavat vaihtoehdot eli lokitiedoston rakenne, muistinhallinta, asetusten tarkistus ja Bluetooth-testiapplikaatiot sekä testisetin toiminta.

5.2 Lokitiedoston rakenne

Kaikista testeistä saadaan ja pitääkin saada lokitiedosto, josta nähdään testin tulokset. Lokitiedostot tulevat xml-muodossa ja seuraavassa esimerkissä nähdään Bluetoothin stressitestin lokitiedosto. Kaikissa testeissä lokitiedostot ovat rakenteeltaan samanlaisia.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="test-x.xsl"?>
<Results>
<TestResult>
<TestApp>BT_Tester</TestApp>
<StartTime>12-09-2005 20:17:11</StartTime>
<EndTime>12-09-2005 20:21:40</EndTime>
<Duration>00:04:29</Duration>
<InputData></InputData>
<Result>PASS</Result>
<Severity>Minor</Severity>
<Details>

Stress Test:
Testi
Found service
Connected
1  Sending data
2  Sending data
3  Sending data
4  Sending data
5  Sending data
```

```
6   Sending data
7   Sending data
8   Sending data
9   Sending data
10 Sending data
Sending time was  :04:22.406250
Average data transfer rate was
19,05442 kB/second
Disconnecting
Disconnected

</Details>
<OutputData>
10 TestFileStress.txt has been sent
</OutputData>
</TestResult>
</Results>
```

Lokitiedostosta löytyy siis testiapplikaation nimi, joka on myös lokitiedoston nimi. Siitä löytyy myös testin alkamis- ja loppumisaika sekä testin kesto. Tietysti siitä löytyy myös tärkein tieto eli se, onko testi mennyt läpi. Yksityiskohdissa (Details) on kyseisen testin suoritukseen liittyviä tietoja, kuten esimerkiksi tässä testissä lähetysaika ja keskimääräinen siirtonopeus.

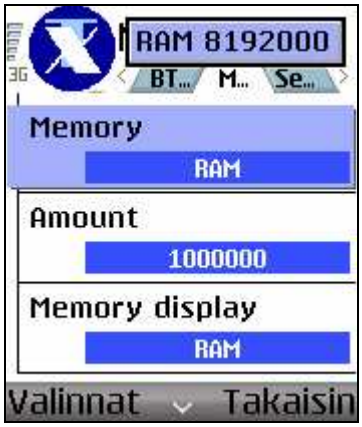
Lokitiedoston voi testin suorituksen jälkeen lähettää vaikkapa Bluetoothilla tietokoneeseen, josta sitä on parempi katsoa kuin matkapuhelimesta. Suoritetuista testeistä saadut lokitiedostot ovat liitteinä (1 - 6).

5.3 Muistihallintatestiapplikaatio

Muistinhallinta-applikaation perusnäkymä nähdään kuvassa 10. Kuvassa näkyy, mitä valintoja voidaan tehdä. Muisti-kohdassa (memory) voidaan valita muistikortti, RAM-muisti tai c-asema. Tämän jälkeen voidaan muokata valitun muistin asetuksia. Muistin näyttö -kohdassa (memory display) voidaan valita, näytetäänkö muistikortin, c-aseman, RAM:in, kaikkien näiden tai ei yhdenkään näistä tila tällä hetkellä. Määrä-kohtaan (amount) syötetään haluttu käyttöön jäävä muistinmäärä. Lisäksi on vielä yksi valikko eli lokitiedoston kirjoitus (log writing), jonka saa päälle tai pois.

Kun suoritetaan muistin syönti (eat memory) tai vapautus (free memory), kirjoitetaan siitä lokitiedosto, josta nähdään, miten se onnistui. Lokitiedostot muistin syönnistä ja vapautuksesta ovat liitteessä 1.

Kuvassa 11 on valittu muistin syönti, jonka suorittamisen jälkeen jäljelle jää vain haluttu määrä muistia, joka tässä tapauksessa on 10000 tavua. Kuvassa 12 nähdään, että RAM-muistia on syönnin jälkeen jäänyt 94208 tavua eli hieman vähemmän kuin haluttiin. Ero halutun ja saavutetun muistin määrän välillä on niin pieni, ettei sillä ole merkitystä. Samassa kuvassa on nyt valittuna RAM-muistin vapautus. Kuten kuvassa 13 nähdään, muistia on vapautunut runsaasti käyttöön.

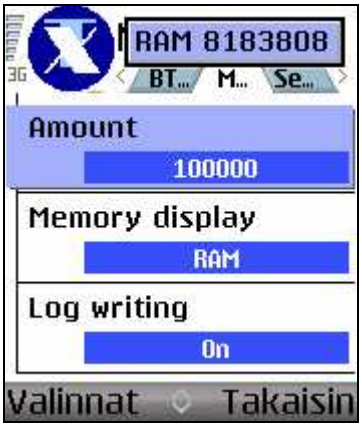


Kuva 10. Muistinhallinnan perusnäkymä

Kuva 11. Muistin syönti

Tämä applikaatio on erittäin hyödyllinen silloin, kun jossain tietyssä testissä tarvitaan paljon muistia. Applikaatio on hyödyllinen myös rasitustestauksessa, jolloin puhelimeen saadaan helposti aikaiseksi ylimääräistä rasitusta, kun vähennetään käytössä olevan muistin määrää. Muistia syömällä pystytään myös tutkimaan miten matkapuhelin toimii, kun muistia ei ole paljoa käytössä.

Kuva 12. Muistin vapautus



Kuva 13. Muistin määrä vapautuksen jälkeen

Näin muistinhallinta-applikaatiota voidaan paremminkin kutsua testauksen apuvälineeksi kuin omaksi testiapplikaatioksi.

5.4 Asetusten tarkistustestiapplikaatio

Kuvassa 14 on applikaation päänäkymä. Tässä ei voida muokata mitään asetuksia.

Kuvassa 15 nähdään eri vaihtoehtoja, joita tällä applikaatiolla voidaan tehdä. Nämä ovat: tallettaa ja vertailla asetuksia sekä suorittaa testi. Testin suorittaminen palauttaa matkapuhelimeen alkuperäiset asetukset ja käynnistää puhelimen uudelleen.

Kun testissä pitää muuttaa paljon asetuksia tietynlaisiksi, tällä applikaatiolla voidaan kopioida oikeanlaiset asetukset tiedostoon. Seuraavalla kerralla, kun ryhdytään suorittamaan samaa testiä ja asetukset on muutettu todennäköisesti oikeiksi, voidaan tehdä asetusten vertailu. Vertailu kirjoittaa lokitiedoston, jossa näkyy, onko asetuksissa vielä eroja. Suoritetusta asetusten vertailusta on lokitiedosto liitteessä 2. Toinen hyöty on se, että jos halutaan varmasti saada alkuperäiset asetuk-

set päälle, niin ajetaan testi, jolloin testi tekee samalla lokitiedoston ennen alkuperäisten asetusten palauttamista olevista asetuksista sekä uusista alkuperäisistä asetuksista. Näin voidaan helposti varmistaa, että asetukset ovat todellakin alkuperäiset.

Myös tätä applikaatiota voidaan kutsua paremmin testauksen apuvälineeksi kuin varsinaiseksi testiapplikaatioksi.

Kuva 14. Applikaation perusnäkymä

Kuva 15. Applikaation valikko

5.5 Bluetooth-testiapplikaatio

Bluetooth-applikaatio itsessään sisältää neljä eri testiä, jotka ovat laitteiden etsintä, aktivointi-, tiedostonsiirto- ja stressitesti. Näihin testeihin vaaditaan kaksi matkapuhelinta, joissa on Bluetooth. Seuraavaksi esitellään nämä testit.

5.5.1 Laitteiden etsintä (Service search)

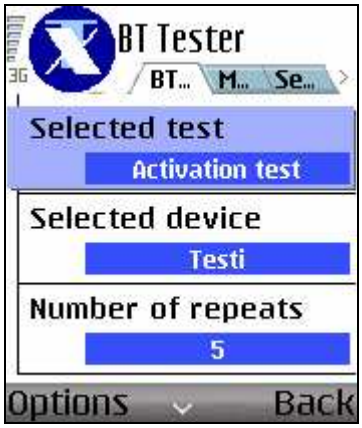
Kuvassa 16 on laitteiden etsinnän perusnäkymä, josta näkyy seuraavat valikot: valikoitu testi (selected test), valikoitu laite (selected device) ja etsi kaikki laitteet (scan all devices). Testivalikosta voidaan siis valita, mikä Bluetooth-testi halutaan suorittaa. Laitteen kohdalle voidaan valita jokin tietty matkapuhelin, jota testissä halutaan käyttää tai jättää kohta tyhjäksi. Etsi kaikki laitteet -kohtaan voidaan valita kyllä tai ei. Tähän kannattaa valita ei, jos on valinnut jonkin tietyn laitteen, sillä näin ei kulu turhaan aikaa kaikkien laitteiden etsintään.

Tällä testillä on erittäin helppoa ja nopeaa etsiä kaikki Bluetooth-laitteet. Samalla varmistutaan siitä, että laitteiden etsintä toimii, mikä on eräs Bluetoothin perusominaisuus. Liitteessä 3 on lokitiedosto laitteiden etsintä testistä.

Kuva 16. Laitteiden etsinnän perusnäkymä

5.5.2 Aktivointitesti (Activation test)

Kuvassa 17 on aktivointitestin perusnäkymä, josta nähdään valittu testi (selected test) ja laite (selected device) sekä toistojen määrä (number of repeats). Kuvasta puuttuu yksi valikko, joka on kaikkien laitteiden etsintä (scan all devices).



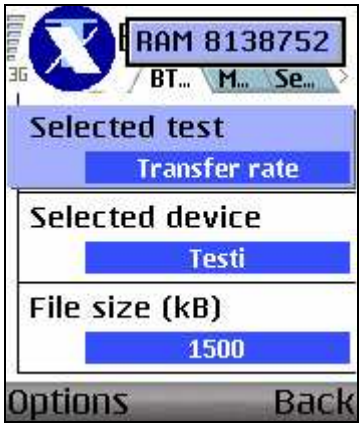
Kuva 17. Aktivointitestin perusnäkymä

Tällä testillä testataan Bluetooth-yhteyden avaamista ja sulkemista. Toistojen määrä -valikolla kerrotaan, kuinka monta kertaa yhteys avataan ja suljetaan. Tällä testillä saadaan selville, kuinka luotettavasti yhteyden muodostus ja sulkeminen toimivat.

Liitteessä 4 olevassa lokitiedostossa tämä testi ei ole mennyt läpi, mikä johtuu siitä, että testiapplikaatiota oli päivitetty ja siihen oli tullut virhe. Kyseisestä lokitiedostosta nähdään, miten testissä tapahtuva virhe ilmoitetaan.

5.5.3 Tiedonsiirtotesti (Transfer rate)

Tällä testillä siirretään halutun kokoinen tiedosto toiseen matkapuhelimeen Bluetoothin kautta. Kuvassa 18 on tiedonsiirtotestin perusnäkymä. Siinä ovat valittu testi (selected test) ja laite (selected device) sekä tiedoston koko kilotavuina (file size (kB)). Tiedoston kokoon syötetään lähetettävän tiedoston koko. Tiedoston maksimikoko on 10000 kilotavua eli 10 megatavua. Testistä tuleva lokitiedosto kertoo tiedoston lähettämiseen kuluneen ajan ja keskimääräisen tiedonsiirtonopeuden. Liitteessä 5 on tiedonsiirtotestin lokitiedosto.

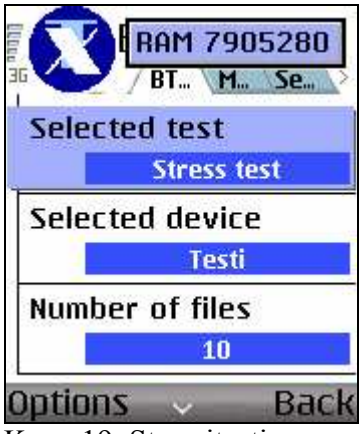


Kuva 18. Tiedonsiirtotestin perusnäkymä

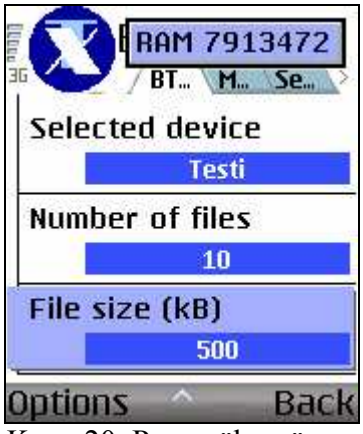
Yksi tämän testin parhaita ominaisuuksia on tiedoston koon helppo valikointi. Testi generoi oikeankokoisen tiedoston ja näin testaajan ei tarvitse etsiä, mistä löytyisi sopivan kokoinen tiedosto. Varsinkin silloin tästä on hyötyä, jos tiedoston pitää olla juuri tietyn kokoinen.

5.5.4 Stressitesti (Stress test)

Stressitesti on hyvin samanlainen kuin edellinen tiedonsiirtotesti, paitsi että testissä lähetetään useita tiedostoja. Kuvassa 19 on stressitestin perusnäkymä ja kuvassa 20 loppuosa tästä näkymästä.



Kuva 19. Stressitestin perusnäkymä osa 1



Kuva 20. Perusnäkymä osa 2

Tästäkin testissä ovat samat perusvalikot: valittu testi (selected test) ja laite (selected device). Nyt myös tiedostojen määrä (number of files) ja tiedostojen koko ovat valittavissa. Tällä testillä saadaan hyvin rasitettua Bluetoothin tiedonsiirtoa. Liitteessä 6 on stressitestin lokitiedosto.

5.6 Testisetin ominaisuudet

Testisetin perusnäkymä näkyy kuvassa 21, kun yhtään testisettiä ei ole luotu. Kuvassa 22 on sama näkymä, kun on luotu pari testisettiä.

Kuva 21. Testisetin perusnäkymä

Kuva 22. Kaksi testisettiä luotu

Testisettiin voidaan valita haluttu määrä testejä, jotka suoritetaan. Kuvassa 22 testisetteihin on valittu ainoastaan Bluetooth-testejä. Bluetooth-testeistä voidaan valita testit, jotka suoritetaan, kuten esimerkiksi aktivointi- ja stressitesti.

Testisetti-ominaisuudesta saadaan suurin hyöty silloin, kun suoritettavia testejä on paljon. Tällöin voidaan valita testit, jotka halutaan ajaa ja laittaa ne ajoon. Samalla testaaja voi tehdä muita töitä tai suorittaa manuaalisia testejä. Testisetti ottaa testien asetukset sellaisinaan, kuin ne on jokaiselle testille asetettu.

6 YHTEENVETO

Tämän insinöörityön tavoitteena oli tutkia, mitä kaikkia regressiotestejä Test-X-työkalulla pystyttäisiin automatisoimaan. Aihetta ryhdyttiin tutkimaan kirjallisuudesta, josta löytyikin loputon määrä tietoa testauksesta ja automatisoinnista. Näistä sekä Series 60 -ohjelmistoalustan ominaisuuksista kerrottiin työn alkupuolella. Viimeisessä luvussa 5 esiteltiin tähän yhtä mahdollista ratkaisua eli Test-X-työkalua. Tämän työn tuloksena saatiin melko paljon automatisoitua regressiotestisettiä, mutta myös rasitus- ja suorituskykytestausta.

Test-X-työkalua tehtiin osittain samaan aikaan, kun myös testiapplikaatioita koodattiin. Tämä hidasti jonkin verran applikaatioiden tekoa, koska itse Test-X oli vielä kesken eikä toiminut kovin vakaasti. Kun se saatiin valmiiksi, myös applikaatioiden koodaaminen nopeutui, kun pystyttiin testaamaan applikaatioita koodauksen yhteydessä. Test-X on helppo asentaa ja käyttää, sekä lisäksi se toimii luotettavasti. Nämä ovat erittäin tärkeitä ominaisuuksia. Puutteitakin löytyy, kuten esimerkiksi tulosten automaattinen siirtäminen tietokantaan. Tämä olisi hyödyllinen ominaisuus, koska se nopeuttaisi testitulosten raportointia. Testisetissä olevia testejä tai eri testisettejä pitäisi pystyä suorittamaan samanaikaisesti. Tällä saataisiin kunnon rasitustestausta matkapuhelimeen, sillä onhan käyttäjälläkin usein monta applikaatiota matkapuhelimessa päällä samanaikaisesti. Tällä hetkellä on siis vain mahdollista suorittaa testit perätysten. Olisi hyvä olla myös erillinen listaus suoritetuista testeistä, jotka ovat menneet ja jotka eivät ole menneet läpi. Tällöin nähtäisiin nopeasti ja helposti testin lopputulos. Jos testi ei mene läpi, sen lokitiedostoon pitäisi tulla ilmoitus, missä testin kohdassa virhe tapahtui. Paras informaatio olisi, jos nähtäisiin testikoodin rivinumero, jossa virhe tapahtui.

Aloitetaan muistinhallintatestistä, josta saadaan paras hyöty stressitestauksessa, kun jätetään muistia vain vähän jäljelle ja siten rasitetaan puhelinta. Muistin vapautuksella ei ole paljoakaan käyttöä testauksessa.

Asetusten tarkistustesti on toteutetuista testeistä huonoin. Siitä ei ole apua juuri nimeksikään. Alkuperäisten asetusten palautus on ainoa ominaisuus, jolle on vähän käyttöä. Asetusten tallennusominaisuus olisi todella hyvä, jos olisi mahdollisuus myös ladata tallennetut asetukset puhelimeen. Se tekisi tästä testistä erittäin hyödyllisen ja säästäisi monelta vaivalta.

Bluetooth-testin automatisointi onnistui todella hyvin. Bluetoothiin saatiin myös rasitus- ja suorituskykytesti automatisoitua. Bluetooth-testien suoritusaika väheni ja testien suorittaminen helpottui huomattavasti.

Muita testejä, joita automatisoitiin, olivat mm. IrDA- ja USB-testit, jotka ovat hyvin samankaltaisia kuin Bluetooth-testit. Näillä saavutettiin myös paljon hyötyä. Kameran testausta saatiin automatisoitua eli kuvan ja videokuvan ottamista. Stressitestausta kameran automatisointi helpotti todella paljon, mutta muuten kameran automatisoinnista ei ajallisesti ollut mitään hyötyä. MMS- ja SMS-viestien lähetys ja vastaanotto automatisoitiin myös. Tässä saavutettiin ajallista hyötyä, kun viestit generoituvat automaattisesti, eikä viestejä tarvitse kirjoittaa itse. Muistikortin ja c-aseman testauksen automatisointi onnistui kaikkein parhaiten. Näihin saatiin yhteensä 20 testiä, joilla testataan muistikortin ja c-aseman perusominaisuuksia, kuten formatointia, tiedoston luontia ja tuhoamista sekä tiedostonsiirtoa. Myös yksi näyttötesti onnistuttiin automatisoimaan. Näyttötestejä on melko hankala automatisoida, koska testaajan pitää yleensä nähdä, mitä näytöllä tapahtuu. Edellä mainitut testit saatiin siis automatisoitua Test-X-työkalulla. Seuraavaksi esitellään, mitä testejä vielä tutkitaan ja mahdollisesti yritetään automatisoida.

Näyttötestejä siis automatisoidaan lisää. Matkapuhelimella soittamista saatetaan yrittää automatisoida osittain, mutta soittaminen manuaalisesti on helppoa ja testaajan pitää kuitenkin puhua ja kuunnella, että ääni kuuluu. Sama koskee videopuhelua, jossa pitää vielä myös näytöltä katsoa, että kuva näkyy. Todennäköisesti näitä ei tulla automatisoimaan. Matkapuhelimen ääni- ja musiikkiominaisuuksien, kuten FM-radion, testaamista automaattisesti tutkitaan, mutta tässäkin on se ongelma, että testaajan pitää kuulla ääni. Mahdollisesti testejä voidaan kuitenkin jonkin verran automatisoida ja näin kenties helpottaa testausta. Seuraavaksi automatisoitavat testit ovat hyvin todennäköisesti Internetissä tapahtuvaa sivujen katselua ja tiedonsiirtoa koskevia testejä. Nämä ovat erittäin tärkeitä ominaisuuksia matkapuhelimessa. Siinä olivat tutkinnassa ja automatisoinnissa olevat testit. Myös olemassa olevia testejä automatisoidaan lisää ja testejä kehitetään.

Testauksen automatisointi onnistui kaiken kaikkiaan hyvin. Matkapuhelimen järjestelmätestaukseen kuuluu paljon testattavia ominaisuuksia ja kaikkia testejä on mahdotonta automatisoida. Tällä työkalulla saadaan melko paljon automatisoitua testejä, mutta harkinnan arvoista olisi tutkia, olisiko markkinoilla jokin toisenlainen automatisointityökalu. Hyvä lisä testaukseen voisi olla esimerkiksi jokin sentyyppinen työkalu, jonka avulla tietokoneessa olevalla ohjelmalla pystyttäisiin ohjaamaan matkapuhelinta Bluetoothilla ja näin suorittaa testejä. Tällaisella työkalulla saavutettaisiin ehkä lisää automatisointikattavuutta testaukseen.

Paljon kehitystyötä tarvitaan vielä Test-X-työkaluun, jotta siihen saadaan lisää uusia ominaisuuksia. Uusien ominaisuuksien tekeminen testaustyökaluun on jatkuva haaste, koska matkapuhelimiin tulee ominaisuuksia kovalla vauhdilla lisää. Automatisointityökalun tekeminen ei todellakaan ole helppoa ja sen jatkuva kehittäminen vaatii paljon työtä, jotta työkalu pysyisi mukana kehityksessä. Testiapplikaatioiden tekeminen edellyttää jatkuvaa kehitystyötä ja valmiit testiapplikaatiot vaativat ylläpitoa ja kehitystä, jotta niistä saadaan entistä parempia.

Insinöörityön tavoite saavutettiin ja Test-X-työkalun sekä testiapplikaatioiden kehitys jatkuu. Testiapplikaatioita voidaan siis käyttää Series 60 -matkapuhelimien järjestelmätestauksessa. Myös tulevaisuudessa testiapplikaatioita toivottavasti voidaan käyttää suoraan, mutta ainakin niistä saadaan hyvät pohjaratkaisut, mikäli ne eivät suoraan sovellu.

LÄHDELUETTELO

[1] Kit, Edward: Software Testing in the Real World. Addison-Wesley. USA 1995.

[2] Mobile Internet Technical Architecture, MITA: 2. Solutions and tools, IT Press, Nokia, 2002.

[3] Series 60 homepage, http://series60.com/. Series 60, 06.09.2005.

[4] Symbian OS homepage, http://www.symbian.com/. Symbian, 06.09.2005.

[5] Glenford, Myers: The Art of Software Testing. John Wiley & Sons, 1979.

[6] Boris, Beizer: Software Testing Techniques, 2nd edition. International Thomson Computer Press. Boston 1990.

[7] Roger, Pressman: Software Engineering: A Practioner's Approach, Fourth Edition. McGraw-Hill, 1997.

[8] The Computer Museum History Center: 2001 Fellow Awards Banquet: http://www.computerhistory.org/archive/program.pdf, 13.10.2005

[9] Rick, Craig; Stefan Jaskiel: Systematic Software Testing. Artech House Publishers, 2002.

[10] Software Testing Types. http://www.aptest.com/testtypes.html, 08.08.2005.

[11] Mark, Fewster; Dorothy, Graham: Automating Software Testing: Effective Use of Test Execution Tools. Addison-Wesley, 1999.

[12] Elfriede, Dustin; John, Paul; Jeff Rashka: Automated Software Testing: Introduction, Management and Performance. Addison-Wesley, 1999.

[13] Pettichord, Bret: Success with Test Automation, Revised version, June 2001. http://www.io.com/~wazmo/succpap.htm, 22.08.2005.

LIITELUETTELO