VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jokke Kinanen

Testaustyökalun toteuttaminen

Case StarSoft Oy

Liiketalous ja matkailu
2014

VAASAN AMMATTIKORKEAKOULU
Liiketalous ja matkailu

TIIVISTELMÄ

Tekijä	Jokke Kinanen
Opinnäytetyön nimi	Testaustyökalun toteuttaminen
Vuosi	2014
Kieli	suomi
Sivumäärä	44
Ohjaaja	Antti Mäkitalo

Opinnäytetyöni tarkoitus oli rakentaa StarSoft Oy:lle sovellustestaamista helpottava sovellus, jolla voidaan nopeasti käydä läpi haluttu määrä www-osoitteita ja tarkistaa, ettei sovellukselle syötetyllä tunnuksella ole pääsyä sellaisiin www-sivuihin, joihin hänen oikeuksillaan ei pitäisikään päästä.

Wilma URL Pommittaja on ohjelmoitu käyttäen Pascal-ohjelmointikieltä Lazarus ohjelmointiympäristössä.

Opinnäytetyössäni käydään myös hiukan läpi sovellustestaamisen maailmaa ja siihen kuuluvia käsitteitä sekä kaavioita. Tämän lisäksi tekemäni sovelluksen rakentamisesta kerrotaan ja sovellus esitellään. Opinnäytetyössä esitellään myös Wilma URL Pommittajan tulosten vertailuun rakennettu erillinen pieni sovellus.

Avainsanat Ohjelmointi, Pascal, testaus, kehitys, sovellus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Business Economics and Tourism

ABSTRACT

Author	Jokke Kinanen
Title	Developing Testing Software, Case StarSoft Ltd.
Year	2014
Language	Finnish
Pages	44
Name of Supervisor	Antti Mäkitalo

The goal of this thesis was to build software that would help software testers in their jobs at StarSoft Ltd. With this software, a tester can go through masses of www-addresses almost instantly and get valuable data that the user inserted in the software cannot access any data that he/she does not have rights to.

Wilma Url Pommittaja is programmed with Pascal-programming language and it is done using Lazarus IDE.

In this thesis some of diagrams, terms and the world of software testing were examined. Also, the development of Wilma URL Pommittaja was introduced and how it works was described. Still, a second software is introduced, it was made for the thesis process for the comparison of Wilma URL Pommittaja results.

Keywords Programming, Pascal, testing, development, software

SISÄLLYS

KUVIOLUETTELO

MÄÄRITELMÄT JA KÄSITTEET

IP-osoite	Tietokoneen osoite verkossa
Portti	Piste, josta tietokone ottaa vastaan verkon yli tulevaa liikennettä.
HTTP	Internetselainten käyttämä siirtoprotokolla
HTTPS	HTTPS on Salattu muoto HTTP tiedonsiirtoprotokollasta
Pommitus	Pommitukseksi kutsutaan tapahtumaa, kun Wilma URL Pommittaja käy läpi kohde Wilman www-osoitteita.
Lazarus	Avoimen lähdekoodin ohjelmointiympäristö Pascalin ohjelmointiin (Lazarus 2013).
Pascal	Ohjelmointikieli, jolla Wilma URL Pommittaja toteutettiin.
Wilma	StarSoft Oy:n sovellus, joka toimii Primus ja Kurre ohjelmien www-liittymänä.
URL	URL tarkoittaa samaa kuin www-osoite.
MySQL	Ilmainen relaatiotietokanta.
Black Box	Sovellustestaamisen muoto, jossa sovellustestaajalla ei ole pääsyä testattavan sovelluksen lähdekoodiin (Tutorialspoint, 2014).
White Box	Sovellustestaamisen muoto, jossa sovellustestaajalla on pääsy sovelluksen lähdekoodiin (Tutorialspoint, 2014).

1 JOHDANTO

Opinnäytetyönäni oli rakentaa StarSoft Oy:n Wilma ohjelmiston tietoturvan testaamiseen työkalu, jolla voidaan nopeasti tarkistaa, onko ohjelmaan syötetyllä käyttäjätunnuksella pääsy ohjelmaan syötettyihin www-osoitteisiin. Käytännössä sovelluksella siis tarkastetaan, ettei esimerkiksi tietyt oikeudet omaavan opettajan oikeudet ole liian suuret ja ettei hän pääse käsiksi sellaisiin sivuihin, joihin hänellä ei pitäisi olla pääsyä.

Wilma URL Pommittajaa lähdettiin suunnittelemaan noin kaksi vuotta sitten. Tarkoituksena oli saada sovellustestaajien avuksi heidän työtään osaksi helpottava yksinkertainen työkalu.

Sovelluksen kehitys toimii myös hyvänä mittarina itselleni omista ohjelmointitaidoistani. Mielestäni tämä opinnäytetyössä esiteltävä sovellus oli osaltaan haastava ja monimutkainen rakentaa.

Suunnitelma oli toteuttaa sovellus työpöytäsovelluksena, joten ohjelmointikielenä päädyttiin käyttämään jo yrityksessä muutenkin tuttua Pascalia ja ohjelmointiympäristönä avoimen lähdekoodin Lazarusta.

Opinnäytetyössä käydään myös läpi testauksen sovellustestaamiseen liittyviä käsitteitä ja kaavioita. Kaikissa tehdyissä kaavioissa on käytetty esimerkkinä StarSoft Oy:n Wilma -sovellusta.

1.1 Opinnäytetyön tavoitteet ja tutkimuskysymykset

Sovelluksen kehittämisen tavoitteena oli helpottaa StarSoft Oy:n sovellustestaajia käymään läpi suuria määriä Wilman www-osoitteita ja saamaan näiden palauttamasta tiedosta helposti luettavaa informaatiota. Tavoitteena oli myös edistää omaa kokemusta Pascal-ohjelmointikielellä ohjelmoimisesta sekä harjoitella kyseisen kielen käyttöä tehokkaana keinona työkalujen sekä sovellusten rakentamiseen.

Sovellusta kehittäessä tärkeimpiä tutkimuskysymyksiä olivat:

- Kuinka saataisiin helpotettua ja nopeutettua suuren Wilman www-osoitteiden määrän läpikäyntiä?

- Millainen sovellus kyseisiin tarpeisiin vastaisi? Millaista tietoa sen pitäisi antaa sen käyttäjälle?

- Miten sovelluksen tuottamaa tietoa voidaan helposti käsitellä ja lukea?

1.2 Katsaus yritykseen StarSoft Oy ja sen tuotteisiin

StarSoft Oy on suomalainen koulu- ja oppilaitoshallintoon erikoistunut ohjelmistotalo. Yrityksen perusti laihialainen Pentti Tarpio poikiensa Jarin ja Jarton kanssa vuonna 1987.

StarSoft Oy:n ohjelmistokokonaisuus pitää sisällään kolme ohjelmaa:

- **Primus** on tietokantaohjelma, johon tallennetaan lähes kaikki kouluhallintoon liittyvät tiedot. Näihin tietoihin lukeutuu muun muassa opettajien tiedot, opiskelijoiden arvosanat ja muut tiedot sekä opetussuunnitelmat ja tarjonta (StarSoft Oy, 2012).

- **Kurre** on ohjelma työjärjestysten suunnitteluun (StarSoft Oy, 2012).

- **Wilma** on www-liittymä, joka käyttää pohjana Primukseen ja Kurreen tallennettua tietoa. Wilman käyttäjiin lukeutuvat opiskelijat, huoltajat, opettaja ja lähes kaikki, jotka ovat mukana kouluhallinnon arjessa. Wilma on suomalaisten koulujen nykyaikainen työkalu (StarSoft Oy, 2012).

StarSoft Oy:n ohjelmat ovat suomalaisessa koulumaailmassa vahvana vaikuttajana. Esimerkiksi yli 90 prosenttia Suomen peruskouluista käyttää StarSoftin tuotteita (StarSoft Oy, 2012).

1.3 Pascal-ohjelmointikieli

Pascal on Niklaus Wirthin kehittämä ohjelmointikieli, joka pohjautuu Algol-ohjelmointikieleen. Pascal kehitettiin 1960-luvun loppupuolella ja se on ollut eri-

tyisen suosittu esimerkiksi opetuskäytössä sen selkeän rakenteen puolesta. (Webopedia).

Tässä opinnäytetyössä on käytetty Pascalista myöhemmin johdettua olio-ohjelmointiin soveltuvaa Object Pascal -johdannaista. Ohjelmat on käännetty ilmaisella FreePascal -kääntäjällä.

Pascaliin päädyin sen takia, että yrityksessä käytetään kyseistä kieltä ohjelmien toteuttamiseen. Tämä oli minulle täydellinen tilaisuus päästä syvemmälle Pascalin maailmaan ja päästä toteuttamaan sellaisia ratkaisuja, joita ei normaalissa omissa sovelluksissa ole tarvittu.

2 SOVELLUSTESTAUS

Tämän opinnäytetyön pääaiheen ollessa sovellustestaustyökalun toteuttaminen on sopivaa kerrata myös sovellustestausta yleiseltä tasolta. Sovellustestaus on kokonaisuutenaan laaja tuotekehityksen osa-alue, joten tässä raapaistaan vain sen pintaa.

2.1 Yleisesti

Sovellustestaamisen yleinen määritteleminen on vaikeaa, koska jokainen yritys käyttää omaa tapaansa sen toteuttamiseen. Sovellustestaukselle ei ole olemassa testausprosessin määrittäviä standardeja.

Sovellustestaukseksi sanotaan sitä tuotekehityksen osa-aluetta, jossa valmiiksi saatu ohjelman osa tai korjaus varmistetaan toimivaksi siihen tarkoitukseen, jota varten se on luotu. Sovellustestaukseen voi liittyä kokonaan uusien ominaisuuksien testaaminen ennen niiden käyttöönottoa tai jo olemassa olevan ominaisuuden virheen korjaus. Sovelluksessa ilmennyttä virhettä kutsutaan IT-kielessä *bugiksi*. Yleensä ohjelmistotaloilla on itsellään työssä muutama sovellustestaaja, mutta jotkin yritykset ovat erikoistuneet sovellusten testaamiseen ja tarjoavat näitä palveluita ohjelmistotaloille.

Sovellustestaaminen vaatii yleensä tarkkaa dokumentaatiota ja suunnittelua. Nämä kaksi tapahtumaa ovatkin sovellustestaamisessa eniten aikaa vieviä prosesseja ja niiden toteuttaminen kunnolla on tärkein asia sovellustestaamisprosessissa. Erittäin tärkeää on myös, että saadut tulokset dokumentoidaan ja analysoidaan tarkasti.

Ikävä kyllä monet yritykset väheksyvät sovellustestauksen tärkeyttä, ja siihen panostetaan liian vähän. Julkisuuteenkin näitä tapauksia tulee vuosittain, kuten esimerkiksi vuonna 2011 tapahtunut VR:n uuden lipunmyyntijärjestelmän totaalisen epäonnistunut käyttöönotto todistaa (Tietoviikko, 2011).

2.2 Sovellustestauksen tavoite

Sovellustestauksen yleinen tavoite on löytää ohjelmassa tai sen ominaisuudessa esiintyvät virheet ennen kuin ohjelma päätyy maksavien asiakkaiden haltuun. Hyvin toteutettu ohjelman testaaminen vähentää asiakkaiden huomaamia virhetiloja ja täten edesauttaa ohjelmistotaloa luomaan tuotteen, joka tyydyttää asiakasta varmemmin kuin jo valmiiksi toimitettaessa rikkinäinen ohjelmisto.

Sovellustestauksella pystytään myös haluttaessa suorittamaan jonkun tyyppistä laadunvalvontaa kohteena olevalle sovellukselle. Voidaan esimerkiksi kerätä suuria statistiikkoja ohjelman sisältämistä *bugeista*, ja näin saadaan raakaa tilastoa siitä, kuinka laadukasta sovelluksen lähdekoodi on tuolla hetkellä.

2.3 Yleinen kuvaus testausprosessista

Testausprosessiksi kutsutaan sitä kokonaisuutta, jonka testaaja käy läpi esimerkiksi sovelluksen uutta ominaisuutta testattaessa. Testausprosessin rakenne saattaa erota huomattavasti eri yritysten käytössä, koska jokaiselle erilaiselle yrityksen toimintamallille sopii erilainen prosessin kulku.

Seuraavissa kappaleissa käydään läpi yhdenlaisessa testausprosessissa käytäviä vaiheita sekä käytettäviä dokumentteja.

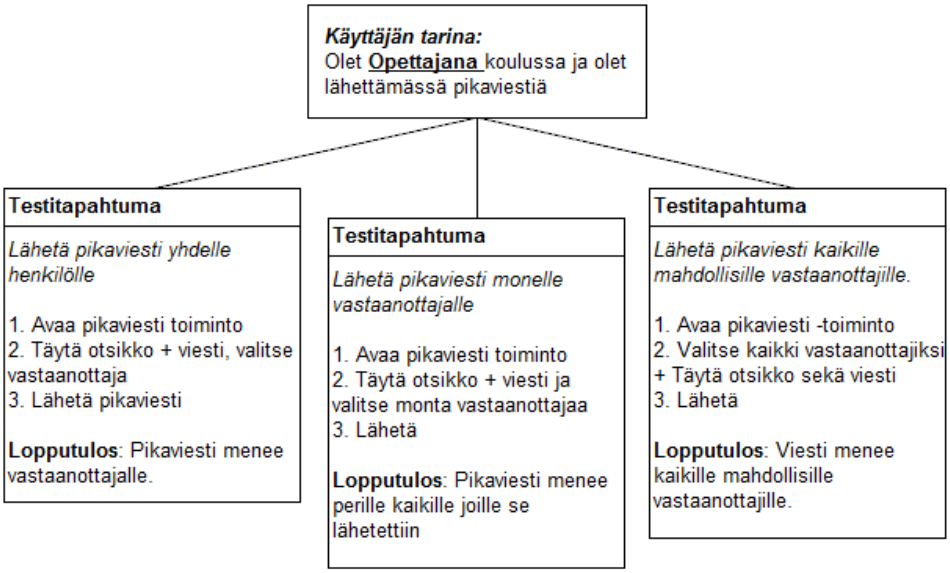
2.3.1 Testaussuunnitelma

Testausprosessi lähtee usein siitä, että sovellustestaaja luo ominaisuuden testaamista kuvaavan testaussuunnitelman sen tiedon pohjalta, mitä hän saa ominaisuudesta etukäteen: tähän suunnitelmaan merkitään esimerkiksi oletettavia ongelmakohtia tai itse ominaisuuden ohjelmoijalta saatavia lisätietoja ominaisuuksista. Kuten, jos kyse on uudesta tiedonsiirrosta, testaajalla täytyy suunnitelmaa tehdessä olla tiedossa ne määritykset, joiden pohjalta koko tiedonsiirron tekevä ominaisuus on suunniteltu ja rakennettu (Software Testing Fundamentals, 2014).

2.3.2 Testitapaukset

Koko testausprosessin ydin ovat erilaiset testitapaukset. Testitapaukset suunnitellaan ennen ominaisuuden varsinaista testaamista. Näitä tapahtumia voi yksi ominaisuus sisältää monia, ja niissä pyritään mahdollisimman hyvin luomaan oikeanlaisia tapahtumia, miten joku sovelluksen käyttäjä tulisi kyseistä ominaisuutta käyttämään. Tämä tuo sovellustestaajalle erilaisia haasteita, koska ohjelmia voivat käyttää täysin eritasoiset tietokoneen käyttäjät, ja testaajan täytyy osata huomioida nekin tilanteet, joissa käyttäjä voi jotakin ominaisuutta yrittää käyttää täysin päinvastaisella lailla eikä siten, miten sitä on alunperin tarkoitettu käytettäväksi.

Itse ominaisuuden lopullinen testaaminen tullaan suorittamaan juuri näitä suunniteltuja testitapauksia käyttämällä.



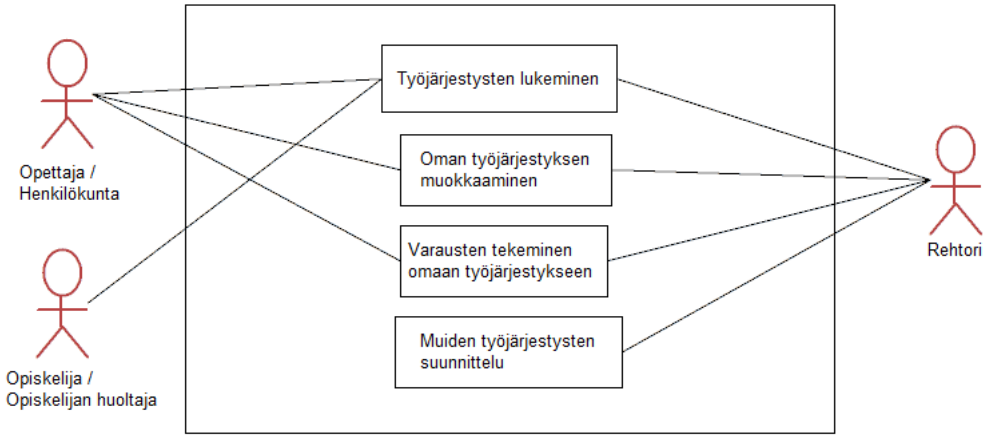
Kuva 2.1. Testitapaus Wilman pikaviestien lähettämisestä (StarSoft Oy, 2014).

2.3.3 Käyttötapaukset eli Use-caset

Käyttötapaukset ovat määrityksiä siitä, miten sovellus toimii esimerkiksi eri käyttäjäryhmien näkymästä. Jotkut käyttäjäryhmät voivat omata enemmän oikeuksia muun muassa tietojen muokkaamiseen, ja joillakin taas saattaa olla samaan tie-

toon oikeus vain tiedon tarkasteluun. Käyttötapauksia esitetään yleensä joko kirjallisessa tai kaaviomuodossa (Techopedia, 2014).

Alla kaksi kaaviota karkean tason use-casesta. Use-caset on tehty StarSoftin Wilma-sovelluksesta.

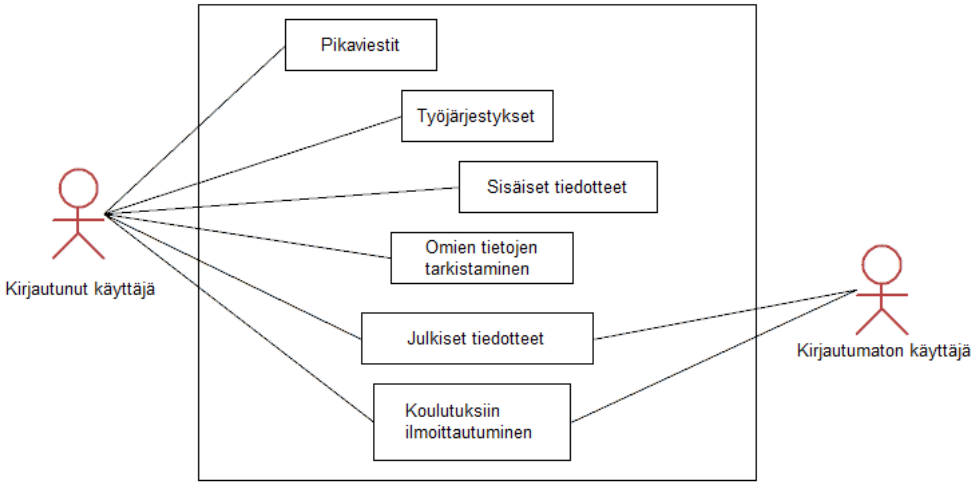


Kuva 2.2. Kuvassa on Wilman työjärjestystoiminto kuvattuna Use-casessa yleisellä tasolla (StarSoft Oy, 2014).

Yllä olevassa kaaviossa (Kuva 2.2) on selvitetty sitä, mitä eri käyttäjäryhmät voivat tehdä Wilman työjärjestys -toiminnolla. Kaavio kuvaa yleiskuvana kyseisen toiminnon toiminnan, mutta ei välttämättä vastaa täysin lopullista tapahtumaa. Wilma-sovelluksenkin tapauksessa on mahdollista rajata suurella määrällä erilaisia oikeuksia se, miten kukakin käyttäjäryhmä (tai jopa käyttäjätasolla) pystyy tekemään muutoksia esimerkiksi työjärjestykseensä.

Oletuksena ovat kuitenkin sellaiset säännöt (esimerkiksi alakoulussa), että opiskelija ja hänen huoltajansa voivat lukea niin omaa kuin opettajienkin työjärjestyksiä. He eivät kuitenkaan pysty muokkaamaan opiskelijan työjärjestystä. Opettajalla taas voi olla oikeus tehdä työjärjestykseensä omia varauksia, esimerkiksi hammaslääkärikäyntiä.

Yleensä koulun rehtori on se henkilö, joka suunnittelee opettajien ja luokkien työjärjestykset ainakin sille tasolle, että opettajat voivat lisätä vain ne oppitunnit, joissa hän itse toimii opettajana.



Kuva 2.3. Esimerkki käyttötapaus Wilmasta (StarSoft Oy, 2014).

Tämän käyttötapauksen (Kuva 2.3) tarkoituksena on siis kertoa siitä, mitä kirjautunut ja kirjautumaton käyttäjä voi tehdä. Wilma itsessään on hyvin suljettu järjestelmä, joten kirjautumattoman käyttäjän tiedonlukumahdollisuudet rajoittuvat vain julkisiin tiedotteisiin, jotka esitetään Wilman kirjautumissivulla.

Koulutuksiin ilmoittautuminen taas on käytössä esimerkiksi aikuisoppilaitosten puolella, joissa Wilma-tunnuksia omaamattomat henkilöt voivat ilmoittautua oppilaitoksen järjestämiin koulutuksiin.

2.3.4 Testausraportti

Testausraportti on testausprosessin tulokset esittävä dokumentti tai dokumenttien sarja. Raportti voi olla hyvinkin yksityiskohtaisesti selvitetty eri vaiheet, joita ominaisuutta testatessa käytiin lävitse. Testausraportit ovat useimmissa yrityksissä virallisia hyvin arkistoitavia dokumentteja, koska niistä voidaan tulevaisuudessa tarkistaa tuliko esimerkiksi asiakkaalla esille myöhemmin tullut ongelma kunnolla testattua sovelluksen testausvaiheessa (Software Testing Help, 2014).

Testausraportti tehdään joko testaussuunnitelmasta tai testitapahtumittain. Testisuunnitelmasta tehtäessä yhdessä laajassa raportissa käydään läpi koko ominaisuuden testaaminen ja siinä heränneet huomiot. Kuitenkin luontavempaa raportti on tehdä testitapahtumittain, jolloin saadaan helpommin yksilöityä testausraport-

tiin juuri ne kohdat, joita kyseisessä testitapauksessa oli suunniteltu läpikäytäväksi (Software Testing Help, 2014).

2.4 Black box ja White box -testausmetodit

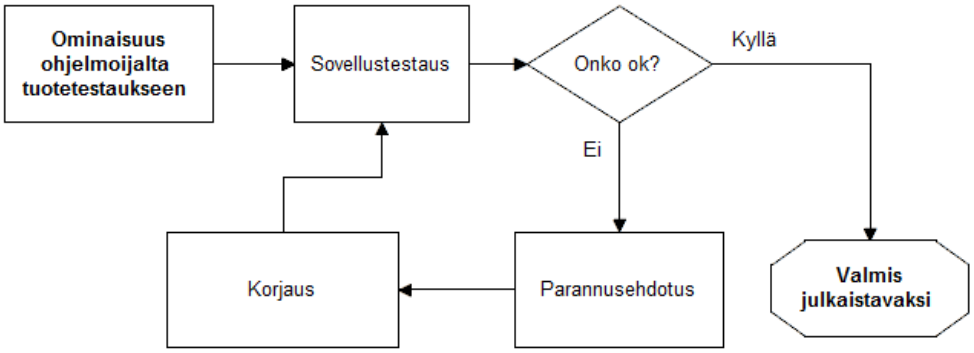
Black- ja White box testaamiset ovat yleisiä sovellustestaamisen metodeja. Erona näissä kahdessa on erilainen lähestyminen testattavaan ominaisuuteen.

Black Box -testaamisessa pääpaino on sovelluksen ominaisuuden testaamisessa niin kuin asiakas sitä tulee käyttämään. Sovellustestaaja yrittää erilaisilla ”in real life” -variaatioilla saada ominaisuudesta esille ominaisuuteen jääneitä virheitä. Black Box testaamista saatetaan suorittaa joissain tapauksissa sellaisilla henkilöillä, joilla ei kyseisen sovelluksen käyttämisestä ole minkäänlaista kokemusta. Tällöin voidaan saada esille sellaisia yksinkertaisia virhetilanteita, joita ei sovelluksen jo tunteva tai ominaisuuden toteuttaja ole tulleet ajatelleeksi tuotekehitysvaiheessa (Tutorialspoint, 2014).

White Box -testaaminen on lähestulkoon Black Box -testaamisen vastakohta. Sovellustestaajilla on vapaa pääsy ominaisuuden ja sovelluksen lähdekoodiin, ja he voivat lähdekoodin avulla rakentaa erilaisia testaustapauksia. Usein White Box -testaajat rakentavat ja suorittavat testitapaukset suoraan sovelluksen lähdekoodissa (Tutorialspoint, 2014).

White Box -testaamisen huono puoli on se, että sovellustestaaja helposti unohtaa ajatella ominaisuutta asiakaslähtöisesti. Tällöin aivan yksinkertaisiakin virheitä saattaa ilmetä ominaisuutta käytettäessä, vaikka testaustapahtumassa sovellus palauttaisikin oikeat arvot käyttäjälle. White Box -testaaminen edellyttää myös sovellustestaajalta käytetyn ohjelmointikielen vankkaa tuntemusta ja kykyä tuottaa kyseistä ohjelmakoodia (Tutorialspoint, 2014).

2.5 Sovellustestaus StarSoft Oy:ssä



Kuva 2.4. Karkea vuokaavio sovellustestauksesta StarSoft Oy:llä (StarSoft Oy, 2014).

StarSoft Oy käyttää sovellustestauksessa Black Box -testaamista muistuttavaa tekniikkaa. Sovellustestaajilla ei itsellään ole pääsyä sovelluksen lähdekoodiin, joten he kokeilemalla erilaisia variaatioita selvittävät, toimiiko kyseinen ominaisuus kuten sen on tarkotuskin toimia. StarSoft Oy:llä tämä eroaa perinteisestä Black Box testaamisesta siten, että sovellustestaajina toimivat jo sovellukset valmiiksi tuntevat henkilöt (StarSoft Oy, 2014).

Yrityksen sovellukset eivät käy läpi asiakkaalla tehtäviä Alpha ja Beta -tason testauksia, vaan sovellukset sovellustestaajien hyväksyminä lähtevät asiakkaiden omasta tahdostaan lataamiin devel-versioihin, joista ne päätyvät myöhemmin julkaistaviin virallisiin versioihin (StarSoft Oy, 2014).

Sovellustestaaja luo ensin testaamastaan ominaisuudesta testaussuunnitelman. Testaussuunnitelmassa käydään läpi yleisesti huomioitavia asioita sekä mahdollisia epäkohtia ominaisuudessa. Tähän pohjaan suunnitellaan erilaisia testaustapahtumia, joilla pyritään saamaan mahdollisimman realistisia tapahtumakuvauksia sovellukseen, ja näiden avulla yritetään löytää mahdollisia virheitä. Kun itse sovellustestaus on suoritettu, tehdään kyseisestä testaustapahtumasta testausraportit, josta ilmenee muun muassa ominaisuudessa ilmi tulleet epäkohdat ja annetut parannusehdotukset (StarSoft Oy, 2014).

Tarkan suunnittelun ja tulosten dokumentoinnin avulla voidaan kyseistä ominaisuutta ja sen testausprosessia tarkastella jälkikäteen tilanteen vaatiessa. Dokumentoinnin tarkoitus onkin mahdollistaa testausprosessin jälkikäteen arviointi ja uudelleen läpi käyminen. Erityisestikin ne henkilöt, jotka kirjoittavat käyttöohjeita käyttäjälle voivat tarvittaessa käyttää näitä dokumentteja ulkoisena tietopankkina (StarSoft Oy, 2014).

Sovellustestaajilla on myös käytössään juuri StarSoft Oy:n omien sovelluksien testaamiseen tehtyjä työkaluja, joiden monipuolisilla ja räätälöidyillä ominaisuuksilla saadaan luotua paljon oikeaa tilannetta muistuttavia tapahtumia ohjelmistoon, ja näin saadaan aikaan mahdollisimman realistisia testausympäristöjä (StarSoft Oy, 2014).

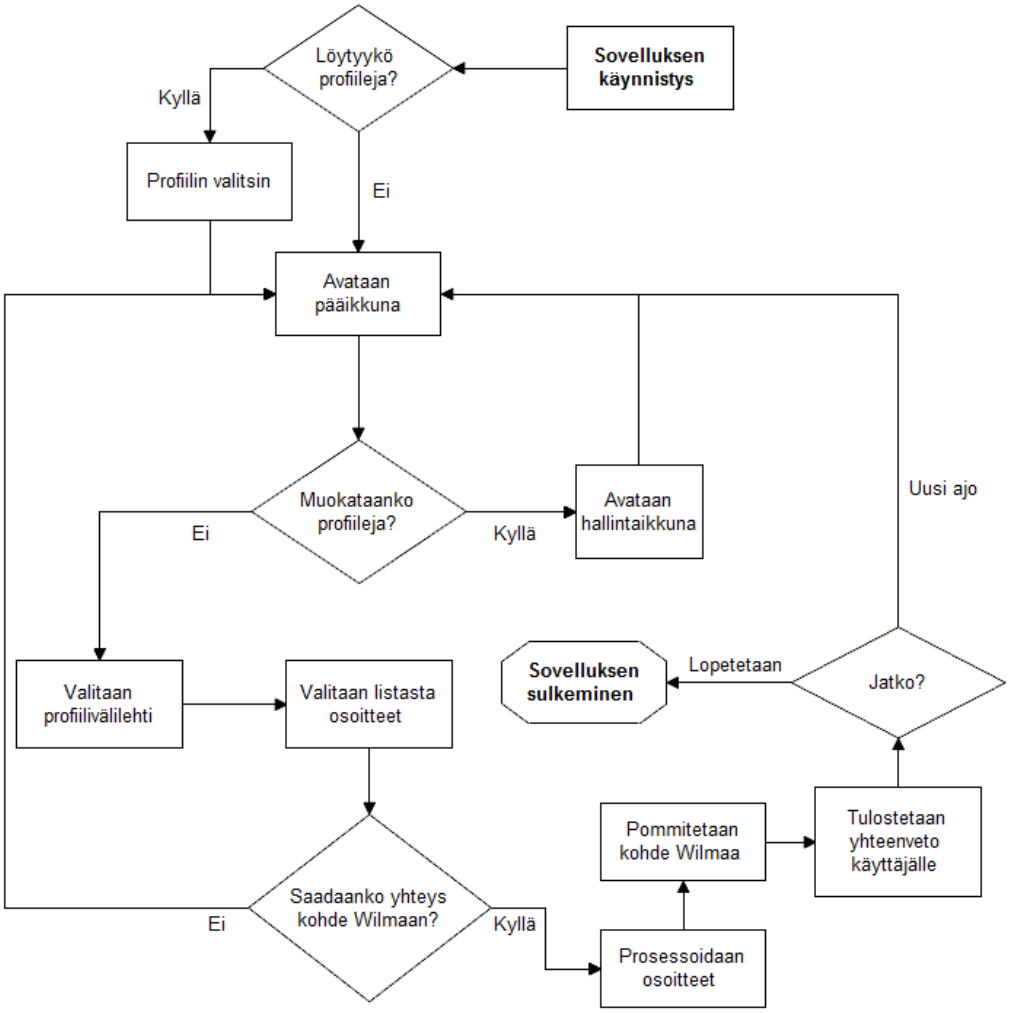
3 WILMA URL POMMITTAJA TYÖKALUNA

Wilma URL Pommittaja on StarSoft Oy:lle testaukseen tehty työkalu. Sen tarkoituksena on käydä läpi sille syötettyä listaa www-osoitteista. Ohjelma kirjautuu annetuilla tunnuksilla sisään kohteena olevaan Wilmaan ja lähettää kyselyn jokaiseen listalla olevaan www-osoitteeseen yksi kerrallaan.

Pommituksesta muodostetaan tässä luvussa myöhemmin selvitettävällä tavalla käyttäjälle helposti luettavaa tietoa. Samassa annetaan käyttäjälle mahdollisuus tallentaa saadusta tiedoista csv-muotoinen tekstitiedosto, jota voidaan käyttää myöhemmin tulosten vertailussa.

Tätä saatua tietoa voidaan käyttää vertailukohtana myöhemmin tehtyyn Wilma URL Pommittajan ajoon ja näin saada tietoa siitä, onko jokin ohjelmistoon tehty muutos aiheuttanut muutoksia käyttäjien oikeuksissa nähdä jotain tiettyä www-sivua tai esimerkiksi sellaisen ryhmän sivua, jota kirjautunut käyttäjä ei itse opeta.

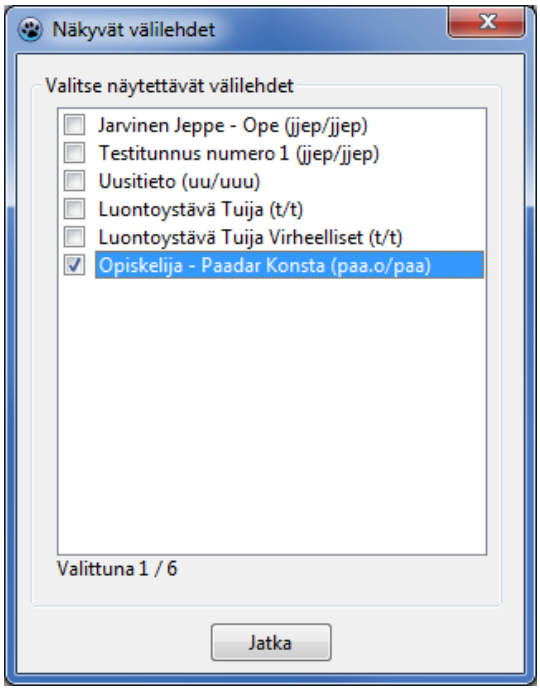
Seuraavalla sivulla vuokaavio Wilma URL Pommittajan toiminnasta (Kuva 3.1).



Kuva 3.1. Vuokaavio, joka kuvaa sovelluksen toimintaa (StarSoft Oy, 2014).

3.1 URL Pommittajan käyttäminen

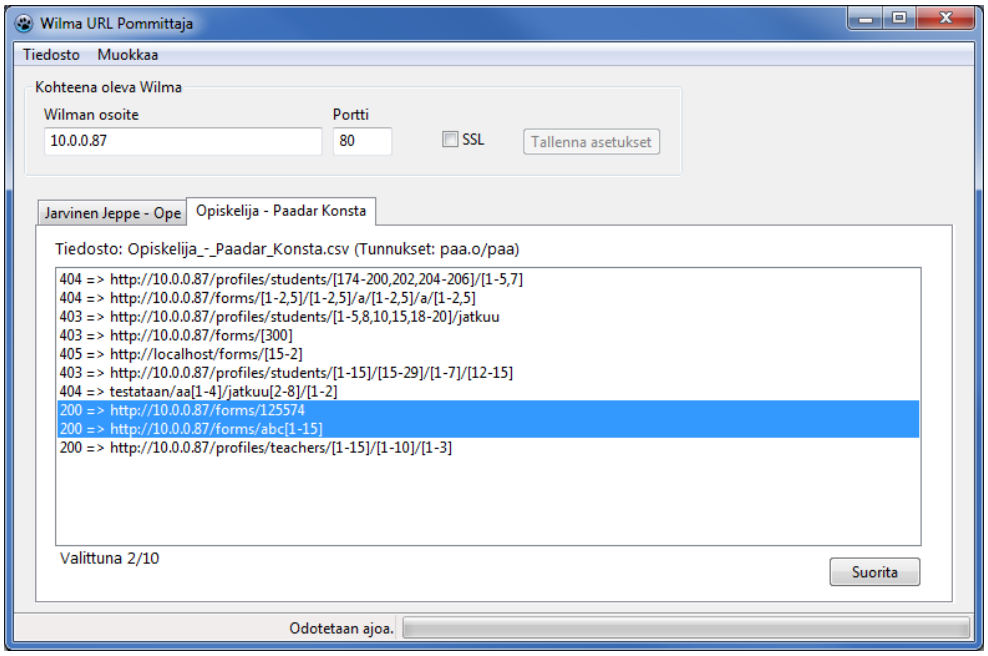
Mikäli Wilma URL Pommittajaan on jo luotu profiileja, käynnistyessään sovellus näyttää käyttäjälle ikkunan, josta hän voi valita pääikkunassa omina välilehtinä näytettävät ohjelmaan tehdyt profiilit. Näkyviä välilehtiä voidaan aina hallita sovelluksen pääikkunan *Muokkaa*-valikosta.



Kuva 3.2. Profiilien valintaikkuna.

Jokainen tämän tyyppinen profiili sisältää Wilman käyttäjätunnuksen ja salasanan lisäksi listan www-osoitteita, joita näillä tunnuksilla halutaan kyseltävän kohde Wilmalta.

Käyttäjän jatkaessa Profiilien valintaikkunasta, aukaisee sovellus hänelle Wilma URL Pommittajan pääikkunan. Pääikkunassa käyttäjälle näytetään hänen valitsemansa profiilit välilehtinä. Käyttäjä voi myös tästä ikkunasta määrittää kohde Wilman IP-osoitteen sekä portin. Kohde Wilman asetukset voidaan tallentaa erilliseen tekstitiedostoon käytettäväksi seuraavilla sovelluksen käyttökerroilla.



Kuva 3.3. Sovelluksen pääikkuna.

Pääikkunan *Muokkaa*-valikon alta löytyy omat toimintonsa niin profiilien kuin www-osoitteidenkin hallintaan.

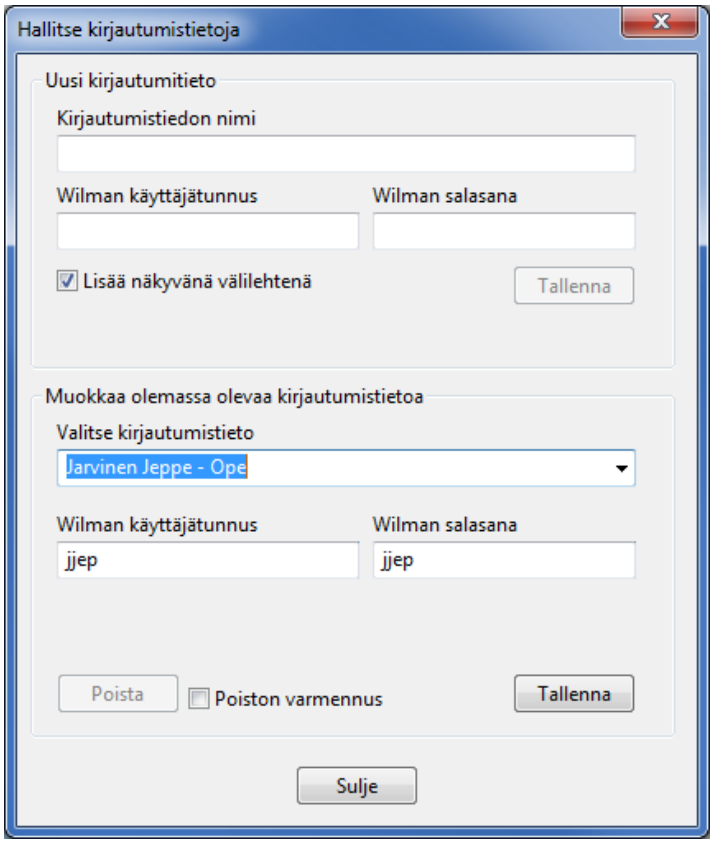
3.1.1 Profiilien hallinta

Profiilien hallintasivulta voidaan hallita sovelluksen käyttämiä käyttäjäprofiileja. Käyttäjä voi tästä näkymästä:

- Luoda kokonaan uusia profiileja, joille määritetään profiilin nimi, Wilman käyttäjätunnus sekä Wilman salasana. Uutta profiilia tehtäessä käyttäjä voi laittaa rastin kohtaan *Lisää näkyvänä välilehtenä*, jolloin profiilien hallintaikkuna suljettaessa käyttäjäprofiilista luodaan oma välilehtensä sovelluksen pääikkunaan.

- Käyttäjä voi muokata jo olemassa olevan profiilin Wilman käyttäjätunnusta ja Wilman salasanaa.

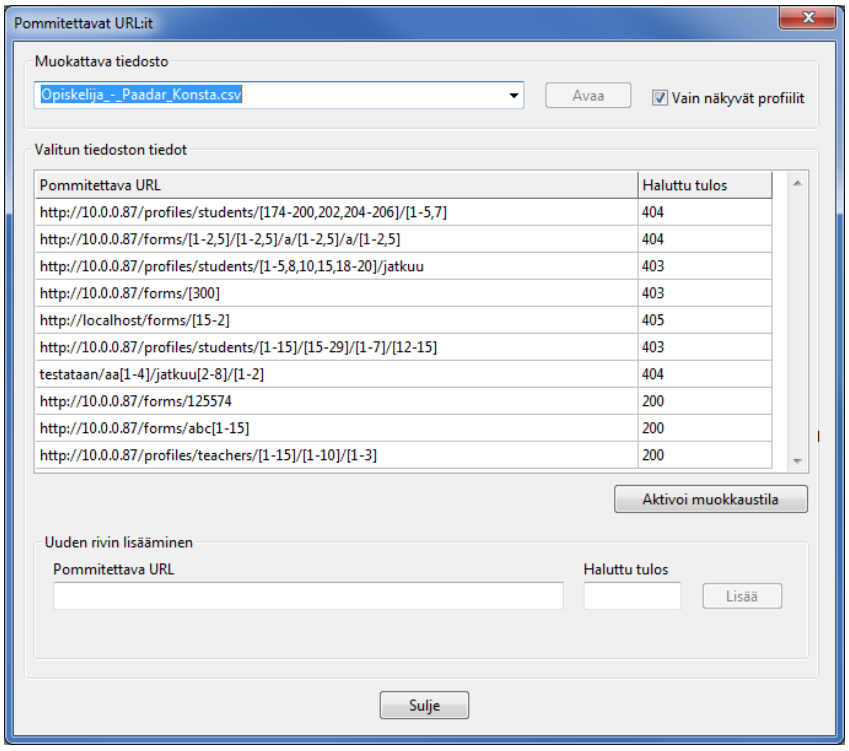
- Poistaa olemassa olevia profiileja. Tämän kanssa käyttäjän kuitenkin täytyy olla varovainen, koska samalla poistetaan myös kaikki tälle profiilille määritetyt www-osoitteet.



Kuva 3.4. Profiilitietojen hallintaikkuna.

3.1.2 Www-osoitteiden hallinta

Www-osoitteiden hallintaikkunaa avattaessa valitaan ensin ikkunan ylälaidasta muokattava profiili. Alasvetovalikon valinnan jälkeen painetaan *Avaa*-painiketta ja *Valitun tiedoston tiedot* -osiossa olevaan taulukkoon päivittyy tuon kyseisen listan osoitteet (Kuva 3.5).



Kuva 3.5. Www-osoitteiden hallintaikkuna.

Jokaista www-osoitetta kohden on kaksi tietoa, jotka käyttäjän täytyy täyttää:

- Pommitettava URL, joka voidaan antaa ilman Wilman osoitetta tai Wilman osoitteen kanssa, sovellus osaa muokata osoitteet juuri kohteena olevaan Wilmaan soveltuviksi. Osoitteeseen voidaan myös lisätä [1-5]-merkinnällä sovellukselle komento käydä sama osoite läpi jokaisella tuolle välille merkityllä numerolla. Esimerkiksi jos sovellukselle syötetään Wilman osoite *http://10.0.0.87/forms/[1-3]*, käy sovellus läpi seuraavat osoitteet:
    - http://10.0.0.87/forms/1
    - http://10.0.0.87/forms/2
    - http://10.0.0.87/forms/3

  Tämä mahdollistaa isomman määrän tiettyä osoiterakennetta käyttävien sivujen läpikäynnin helpommin, ettei käyttäjän tarvitse itse kirjoittaa suurta määrää lähes samaa osoitetta sovellukseen.

- Haluttu tulos, joka annetaan joko HTTP-virhekoodimuodossa (esimerkiksi 404) tai Wilman virhekoodimuodossa (esimerkiksi forms-08, common-404).

Uusia osoitteita käyttäjä voi lisätä listaan ikkunan alapuolella olevasta *Uuden rivin lisääminen* -laatikosta. Käyttäjän painaessa *Lisää*-painiketta rivi ilmestyy automaattisesti taulukon alimmaksi riviksi. Jos käyttäjä haluaa muokata lisäämiään riviä, hänen täytyy painaa *Aktivoi muokkaustila* -painiketta taulukon alareunasta. Tehtyään muutokset suoraan taulukkoon käyttäjä tallentaa muutoksensa painamalla esiin ilmestyvää *TALLENNA MUUTOKSET* -painiketta.

3.1.3 URL Pommituksen suorittaminen

Käyttäjä maalaa pääikkunan laatikosta ne osoiterivit, jotka haluaa ajettavaksi Wilman pommituksessa. Valittuaan rivit käyttäjä painaa *Suorita*-painiketta ja ohjelma aloittaa rakentamalla osoitteista listan. Rakennettua listaa käyttäen sovellus käy jokaisen listalta löytyvän osoitteen läpi ja suorittaa sillä kyselyn Wilmaan. Käyttäjälle tapahtuva prosessi näytetään sekä numeraalisena, läpikäytyjen Wilma www-osoitteiden määränä ja valmistumista kuvaavana valmistumispalkkina.

Kuva 3.6. Valmistumista kuvaava alapalkki.

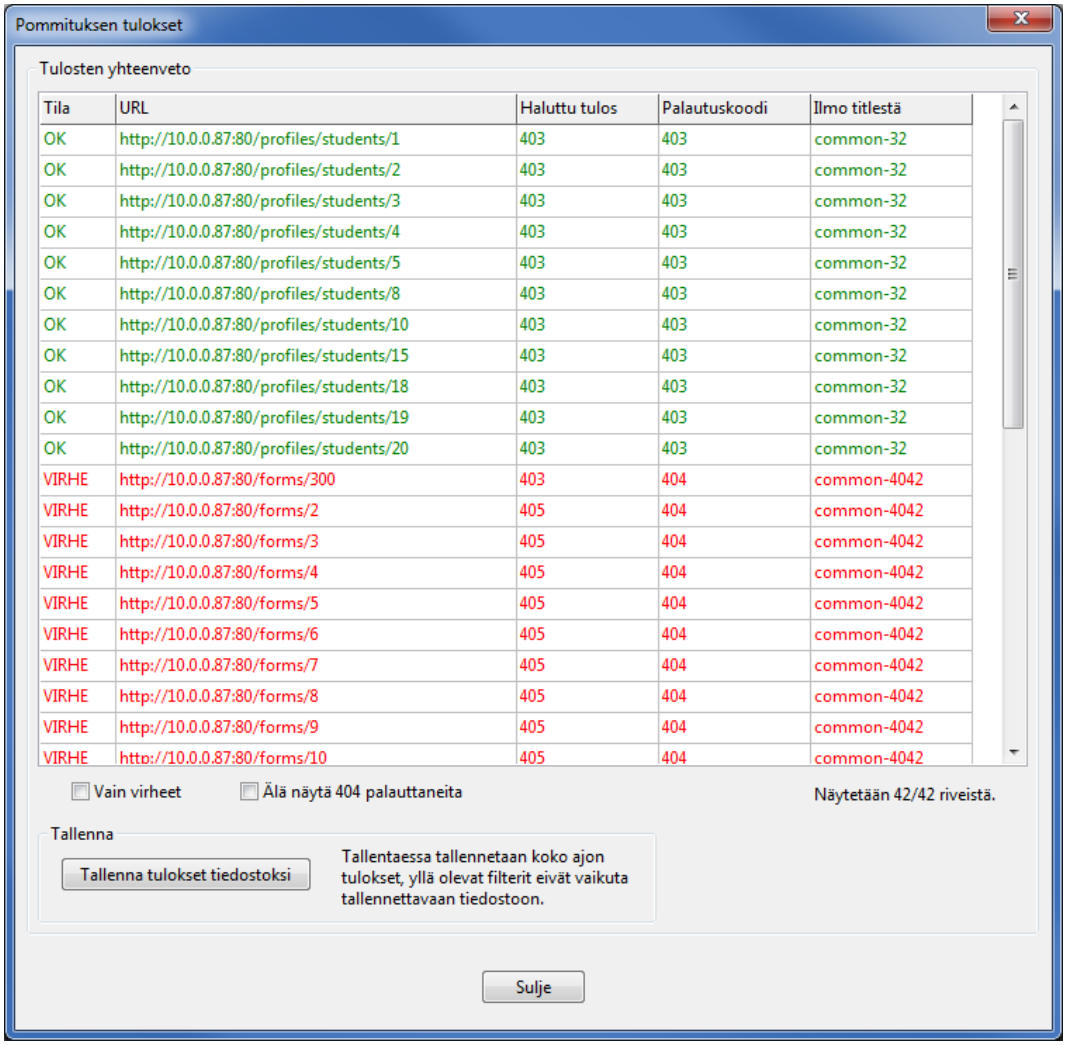
3.2 Pommituksen tulosten lukeminen

Kun kaikki osoitteet on käyty lävitse, avaa sovellus käyttäjälle tulosten lukuun suunnitellun ikkunan. Tässä ikkunassa käyttäjälle näytetään värikoodatussa taulukossa kyselyn tulokset. Taulukko koostuu viidestä sarakkeesta:

- ”Tila”
- ”URL” (www-osoite)
- ”Haluttu tulos”

- ”Palautuskoodi” (Sivun palauttama HTTP-virhekoodi)
- ”Ilmo titlestä” (Wilman HTML-koodista otettu virhekoodi)

Alla on kuva pommitustapahtumasta käyttäjälle näytettävästä tulosikkunasta.



Kuva 3.7. Pommituksen tulosikkuna.

Jokaista osoitetta varten sovellus tekee tähän taulukkoon yhden rivin. Tämän rivin väri on joko *punainen* tai *vihreä*. Jos rivin värinä on *punainen* (*Tila*-sarakkeessa teksti VIRHE), kertoo se käyttäjälle, että rivin *Haluttu tulos* -sarakkeen tieto ei vastaa Wilman antamaa virhekoodia (*Ilmo titlestä* -sarake) eikä HTTP virhekoodia (*Palautuskoodi*-sarake).

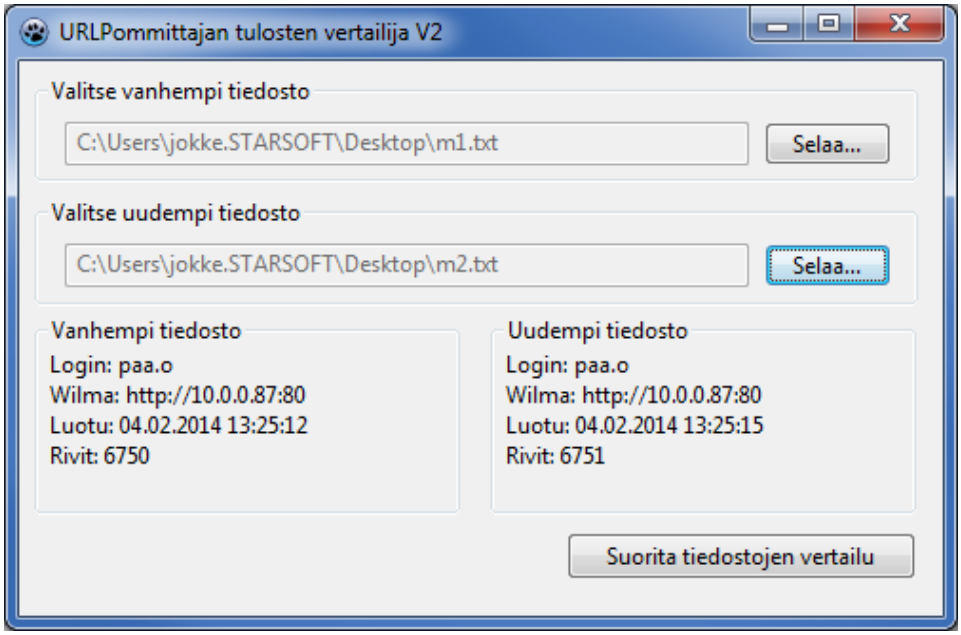
Vihreän värinen teksti tarkoittaa sitä, että rivin *Haluttu tulos* -sarakkeen tieto vastaa joko *Palautuskoodi* tai *Ilmo titlestä* -sarakkeen tietoa.

Tallenna tulokset tiedostoksi -painiketta painamalla käyttäjä voi tallentaa myöhempää käyttöä varten tuloksista csv-tiedoston. Tätä tiedostoa voidaan verrata seuraavan pommituksen tuloksista tallennettavaan samanlaiseen tiedostoon.

Käyttäjä voi sovellusta käyttäessään itse vaikuttaa tulosten selvyyteen. Optimaalinen tilanne sovellukseen käyttöön saadaan, jos www-osoitteita profiilille lisättäessä jokainen osoite palauttaisi halutuksi tulokseksi määritetyn tiedon. Tällöin myöhemmin samalla profiililla pommitusta tehtäessä, kertoo jokainen punaisella värillä tuleva rivi siitä, että jotain on muuttunut edellisten ajojen jälkeen.

4 TULOSTEN VERTAILU

Wilma URL Pommittajan tulosikkunasta tallennettujen tekstitiedostojen vertailemisen helpottamiseksi päätettiin tehdä sovellus, joka pystyy nopeasti käymään tuhansia rivejä tekstitiedostoista ja näyttämään käyttäjälle selkeässä muodossa vertailun tulokset.



Kuva 4.1. Wilma URL Pommittajan tulosten vertailijan pääikkuna.

Vertailu perustuu siihen, että ohjelmalle syötetään kaksi tiedostoa. Näistä vanhempi haetaan *Valitse vanhempi tiedosto* -kohtaan *Selaa*-painiketta painamalla. Ja uudempi haetaan vastaavanlaisesti *Valitse uudempi tiedosto* -kohtaan. Samalla hetkellä kun tiedosto haetaan ja se löytyy, lukee ohjelma sen muistiin ja näyttää käyttäjälle seuraavat tiedot tiedostosta:

- *Login*-tieto kertoo tiedostoa luodessa käytetyn Wilma käyttäjätunnuksen.
- *Wilma*-tiedossa kerrotaan kohteena olevan Wilman www-osoite.
- *Luotu*-tiedossa kerrotaan tiedoston luomisen ajankohta eli se hetki, jolloin pommitus suoritettiin Wilmaan.

- Viimeiseksi kerrotaan käyttäjälle pommitettavien www-osoitteiden määrä kyseisessä tiedostossa.

Tiedostojen vertailu aloitetaan painamalla *Suorita tiedostojen vertailu* -painiketta tulosvertailijan pääikkunasta. Mikäli tässä vaiheessa uudemman ja vanhemman tiedoston *Login* ja *Wilma* -tiedot eivät täsmää toistensa vastaaviin, varmistetaan käyttäjältä onko hän varma, että kyseiset tiedostot ovat keskenään vertailukelpoisia. Mikäli käyttäjä vastaa myöntävästi, aloitetaan tiedostojen sisältöjen vertailu toisiinsa.

4.1 Vertailutapahtuma

Pääikkunan *Suorita tiedostojen vertailu* -painiketta painettaessa on siis molemmat vertailtavista tiedostoista jo luettu tietokoneen muistiin. Sovellus käy uudemman tiedoston viimeisestä rivistä lähtien yksi kerrallaan koko tiedoston läpi. Jokaisen uudemman tiedoston rivin kohdalla sovellus käy vanhemman tiedoston rivit läpi aloittaen tiedoston lopusta.

Miksi aloitetaan tiedoston lopusta eikä alusta? Lopusta aloittamalla saadaan sovelluksen vertailuun käyttämä aika laskemaan huomattavasti. Sovellus poistaa vanhemman tiedoston rivit muistista sitä mukaa, kun niille on löytynyt vastaavuus uudemmasta tiedostosta. Tämä mahdollistaa sen, ettei sovelluksen tarvitse tuhlata turhaa aikaa näiden rivien käsittelyyn seuraavilla kierroksilla. Mikäli tällainen tapahtuma tehtäisiin luettaessa tiedostoja alusta, menisivät tietojen järjestystiedot (*index*) sekaisin, koska olio johon tiedot on luettu, järjestää järjestystiedot uudelleen aina, kun oliota muokataan. Tässä tapauksessa olion muokkaustapahtuma on rivin poisto. Käytetyt rivit poistamalla sovelluksen nopeutta saatiin nostettua todella paljon. Kahden 6500 rivin tiedoston vertaileminen nopeutui jopa 26 000 millisekunnista noin 50 millisekuntiin. Tällainen nopeuden kasvu sovelluksessa on huomattava sen käyttömukavuutta miettien.

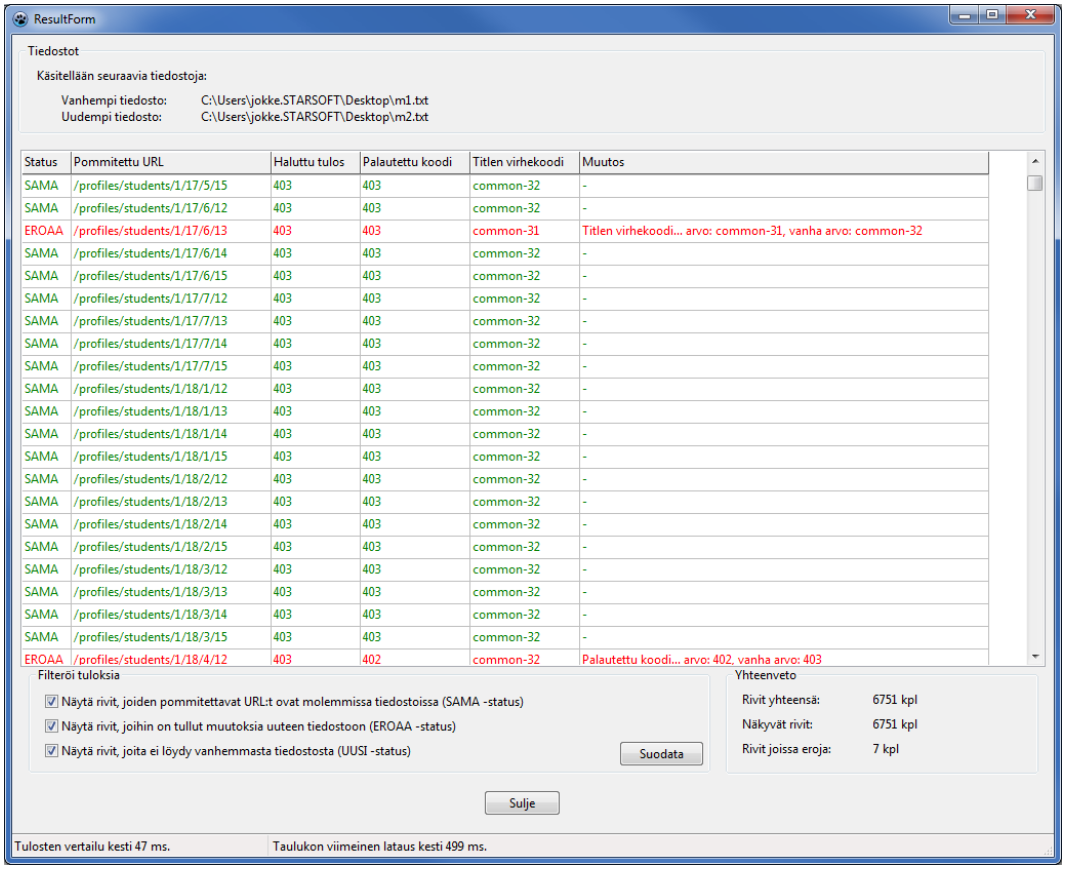
Kun sovellus löytää vanhemmasta tiedostosta täysin täsmäävän rivin, lisää se tulokset-olioon tämän rivin ja siihen lisäksi *Status* tiedon *SAMA*, joka viestittää käyttäjälle siitä, että sama tieto löytyy molemmista tiedostoista.

Jos sovellus ei löydä uudemman tiedoston riviä vanhemmasta tiedostosta, lisää se tulokset-olioon tämän rivin, mutta *Status* tietona *UUSI*.

Mikäli sovellus löytää vanhasta tiedostosta uudessa tiedostossa olevan www-osoitteen, se käy läpi kaikki muut kyseisen rivin tiedot. Jos jokin näistä muista tiedostoista eroaa uuden ja vanhan tiedoston rivien välillä, lisää sovellus *Status* -tiedoksi tiedon *EROAA*. Tämän lisäksi sovellus lisää myös tälle tulosriville *Muutos* tiedoksi muuttuneen sarakkeen sekä sarakkeen uuden ja vanhan tiedon. Esimerkiksi, jos vanhemmassa tiedostossa jokin www-osoite on palauttanut *Palautuskoodin* 403, mutta uudemmassa tiedossa sama osoite palauttaa koodin 200. Tällöin rivin *Muutos*-tiedoksi tulee *Palautettu koodi... arvo: 200, vanha arvo: 403* (Google Inc, 2014).

Kun uudemman tiedoston rivit on päästy läpi, lopetetaan rivien vertailu ja näytetään käyttäjälle yhteenveto vertailun tuloksista.

4.2 Tulosten esitteleminen



Kuva 4.2. Vertailijan tulosten esittelyikkuna.

Käyttäjälle näytetään hyvin paljon Wilma URL Pommittajan tulosikkunan tapainen, värikoodeilla varustettu ikkuna, jossa tulokset näytetään samaan tapaan taulukossa.

Rivit on värikoodattu siten, että käyttäjä erottaa ne helposti toisistaan:

- *SAMA*-rivit on värjätty vihreiksi.
- *EROAA*-rivit on värjätty punaisiksi.
- *UUSI*-rivit on värjätty sinisiksi.

Käyttäjälle on annettu myös mahdollisuus kyseisessä ikkunassa suodattaa näytettäviä tuloksia. Jokaista *Status*-tyyppiä kohden on oma rastikenttänsä, jolla voidaan määrittää näytetäänkö tuota tyyppiä. Suodatus tapahtuu painamalla *Suodata* -

painiketta. Käyttäjälle näytetään myös pientä yhteenvetoa vertailun tuloksista. Yhteenvedossa käyttäjälle kerrotaan rivien lukumäärä, näytettävien rivien lukumäärä sekä niiden rivien lukumäärä, joilla on *Status*-tyyppinä *EROAA*.

Ikkunan alaosassa olevassa osassa käyttäjälle näytetään itse vertailussa kulunut aika sekä viimeisimmän taulukon rakennuksessa kulunut aika millisekunteina. Rakennuksessa kulunut aika päivitetään aina, kun taulukon sisältöä päivitetään, esimerkiksi suodattaessa taulukossa näytettäviä tuloksia.

5 POMMITUSTYÖKALUN TOTEUTUS

Wilma URL Pommittaja on toteutettu Pascal-ohjelmointikielellä Lazarus ohjelmointiympäristössä (Lazarus, 2013). Pascalin käyttöön päädyttiin ohjelmointikielen entuudestaan tuntemisen sekä sen tehokkuuden vuoksi.

Suunnitteluvaiheessa annoin ohjelmalleni muutaman tavoitteen, joita pyrin suunnittelussa sekä sovelluksen toteutuksessa seuraamaan:

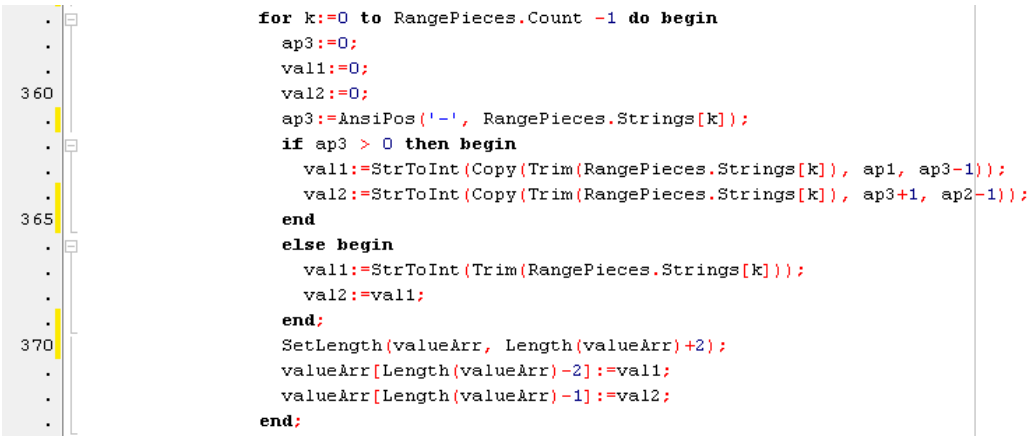
- *Nopeus*, ohjelman tulisi mahdollisimman nopeasti pystyä käsittelemään isoja määriä www-osoitteita.
- *Selkeys*, ohjelman käyttö pitäisi olla mahdollisimman helposti koulutettavissa toisille henkilöille.
- *Tehokkuus*, ohjelman tulisi oikeasti selviytyä täysin sille annettavasta tehtävästä.

Näiden asioiden lisäksi pyrin parhaani mukaan ohjelmoimaan sovelluksen koodin siten, että sitä olisi jälkeenpäin helppo muokata tai laajentaa. Jouduin toteutusvaiheessa kirjoittamaan ohjelman koodin kaksi kertaa kokonaan uudelleen, ja tähän kyllästyneenä päätin tehdä siitä suoraan mahdollisimman helposti jatkettavan.

5.1 Yleisesti sovelluksesta

Sovelluksen päätehtävä StarSoft Oy:n testaustyökaluna on käydä massoittain Wilmaan kohdistuvia www-osoitteita läpi ja kertoa sovelluksen käyttäjälle, millaista tietoa nämä palautuneet sivut palauttavat. Näin voidaan esimerkiksi tietyt oikeudet omaavalla Opettaja-roolilla tarkistaa, ettei hänen roolinsa pääsy tiettyihin Wilman www-osoitteen sisältämiin tietoihin ole muuttunut tehtyjen muutosten jälkeen.

Sovellus koostuu pääikkunasta sekä erillisistä profiilien sekä www-osoitteiden muokkausikkunoista. Näiden lisäksi sovelluksessa on ikkuna, josta käyttäjä voi valita sovelluksessa käytettävät aktiiviset profiilit. Kun itse pommitustapahtuma on suoritettu, käyttäjälle avataan tulokset erillisessä ikkunassa.



```
for k:=0 to RangePieces.Count -1 do begin
  ap3:=0;
  val1:=0;
  val2:=0;
  ap3:=AnsiPos('-', RangePieces.Strings[k]);
  if ap3 > 0 then begin
    val1:=StrToInt(Copy(Trim(RangePieces.Strings[k]), ap1, ap3-1));
    val2:=StrToInt(Copy(Trim(RangePieces.Strings[k]), ap3+1, ap2-1));
  end
  else begin
    val1:=StrToInt(Trim(RangePieces.Strings[k]));
    val2:=val1;
  end;
  SetLength(valueArr, Length(valueArr)+2);
  valueArr[Length(valueArr)-2]:=val1;
  valueArr[Length(valueArr)-1]:=val2;
end;
```

Kuva 5.1. For -silmukka sovelluksen koodista.

5.2 Tietojen tallentaminen

Sovelluksen käyttämät tiedot on tallennettu yksinkertaisiin csv-tyyppisiin tekstitiedostoihin:

- Jokainen sovellukseen lisätty profiili tallennetaan *LOGINS.csv* -nimiseen tiedostoon omaksi rivikseen.
- Jokaista profiilia kohden tehdään oma csv-tiedostonsa, johon tallennetaan tälle profiilille määritetyt pommitettavat Wilman www-osoitteet.
- Ohjelma tallentaa asetuksista oman tekstitiedoston, johon se kirjoittaa kohde Wilman IP-osoitteen, käytettävän portin sekä tiedon siitä, saadaanko kyseiseen Wilmaan yhteys HTTP- vai HTTPS-protokollaa käyttäen.

Wilma URL Pommittaja luo tiedostot, mikäli niitä ei vielä ole olemassa. Tallennushakemistona käytetään sovelluksen juurikansiossa olevaa *data*-kansiota. Profiileja kohti luotavissa, www-osoitteet sisältävissä tiedostoissa käytetään nimenä profiilille annettua nimeä sekä profiilille annettua Wilman käyttäjätunnusta, näin saadaan jokaiselle profiilitiedostolle uniikki nimi, joka tarkistetaan aina uutta profiilia luotaessa mahdollisten samannimisten tiedostojen varalta.

Tietojen tallennus päätettiin suorittaa yksinkertaisiin tekstitiedostoihin siksi, että en nähnyt syytä tuoda ohjelman kylkeen omaa tietokantaa tässä vaiheessa, koska

tallennettavan tiedon määrä on vielä pieni. Jos tallennettavaa tietoa tulee tulevaisuudessa enemmän, täytyy vakavasti harkita erillisen tietokannan, kuten MySQL:n käyttöä.

5.3 Yhteyden ottaminen Wilmaan

Wilma URL Pommittajan pommittaminen perustuu GET- ja POST-kyselyiden lähettämiseen Wilma-palvelimelle. Näiden kyselyiden suorittamiseen käytettiin Pascaliin löytyvää valmista verkkoliikennettä helpottavaa Synapse-kirjastoa. Tämä kirjasto antaa kehittäjälle laajan paketin erilaisia verkkoliikenteeseen liittyviä valmiita toimintoja. Synapse helpottaa muun muassa FTP-, LDAP- ja HTTP-protokollien käyttämistä sovelluksessa. (Ararat Synapse, 2007).

Ararat synapse -kirjasto mahdollistaa myös yhteyden ottamisen SSL-salattuihin web-palvelimiin, jotka siis tunnistaa https-muotoisesta www-osoitteesta. Tämä saadaan helposti aktivoitua lisäämällä kaksi kappaletta dll-sovelluslaajennuksia rakennettavan sovelluksen juurikansioon.

Kirjautuminen Wilmaan sekä uloskirjautuminen Wilmasta suoritetaan *POST*-kyselyitä käyttäen. Kaikki muut kyselyt sovellus lähettää Wilmalle *GET*-kyselyinä. Sovellus kirjautuu Wilmaan sisään vain kerran, eikä uudelleen jokaista kyselyä tehtäessä.

5.4 Pommitettavien www-osoitteiden muodostaminen

Kun käyttäjä painaa sovelluksen pääikkunasta *Suorita*-painiketta, kerää sovellus kentästä valitut osoitteet yhteen ja aloittaa niiden käsittelyn sellaiseen muotoon, että ne vastaavat Wilmassa käytettäviä www-osoitteita.

Sovellusta toteutettaessa www-osoitteiden muodostaminen tuotti suurimman ongelman. Käytettävyyden kannalta olennainen asia oli, että www-osoitteita voidaan käyttää mihin tahansa Wilmaan ilman, että niitä täytyisi jokaista Wilmaa varten kirjoittaa uudelleen. Myöskään ei haluttu suoraan säännöstellä sitä, missä muodossa osoitteet täytyy ohjelmalle syöttää. Mahdollistettiin käyttäjän syöttää osoitteet kolmessa muodossa:

- http://wilma.kunta.fi/teachers/11
- /teachers/11
- teachers/11

Jos kyseinen www-osoite alkaa sanalla *http://* tai *https://*, niin pilkotaan osoite paloihin aina kauttaviivan kohdalta (”/”) ja käsketään sovellusta rakentamaan tällaiset osoitteet uudelleen käyttämällä kohde Wilman asetuksia. Tällöin, jos käyttäjä on syöttänyt osoitteen *http://wilma.kunta.fi/teachers/11* ja kohde-Wilman osoite onkin *https://wilma.kuntaliitoskunta.fi*, muodostaa sovellus kohde-Wilman osoitteesta ja syötetystä www-osoitteesta toimivan kokonaisuuden *https://wilma.kuntaliitoskunta.fi/teachers/11*. Jos taas www-osoite ei itsessään sisällä http tai https -osoitetta, lisää sovellus automaattisesti siihen kohteena olevan Wilman www-osoitteen.

Käyttäjällä on myös www-osoitetta määrittäessä mahdollisuus pistää osoitteisiin mukaan numeroalueita, joita käyttäen sovelluksen halutaan rakentavan yhtä osoitetta käyttäen monta eri osoitetta. Esimerkiksi, jos käyttäjä syöttää seuraavan osoitteen:

- http://wilma.kunta.fi/teachers/[1-2]

luo sovellus tätä riviä käyttäen pommitettavien www-osoitteiden listaan kaksi osoitetta *teachers/1* ja *teachers/2*. Näitä numeroalueita voi kussakin osoitteessa olla maksimissaan 4, joten yhdellä osoitteella voidaan parhaassa tapauksessa ajaa lävitse tuhansia www-osoitteita. Numeroalueessa on myös mahdollista jättää tiettyjä numeroita väliin käyttämällä pilkkua erottimena numeroalueessa. Tällöin osoite on esimerkiksi muotoa:

- http://wilma.kunta.fi/teachers/[1-2,4,6-7]

Yllä mainitusta osoitteesta muodostetaan www-osoitteet *teachers/1*,*2*,*4*,*6* ja *7*. Tämä merkitsemistapa mahdollistaa tilanteet, joissa esimerkiksi opettajan samassa koulussa opettaisivat opettajat korttinumerolla 3 sekä 5, ja näitä opettajia ei halut-

taisi mukaan tähän testaustapahtumaan, koska niihin profiiliin määritetyillä tunnuksilla on esikatseluoikeudet.

Hakasulkeissa olevien ”palojen” läpi käyminen oli sovellusta rakennettaessa haastavaa. Sovellus piti saada luomaan kaikki mahdolliset yhdistelmät arvoista, joita www-osoitteissa on sulkeisiin laitettuna. Sovellus laskee annetuista arvoväleistä, esimerkiksi www-osoitteesta http://wilma.kunta.fi/teachers/[1-2]/[1-2], kaikki mahdolliset numeroyhdistelmät ja tekee www-osoitteen jokaisia numeroyhdistelmiä kohti. Yllä olevasta www-osoitteesta tulee www-osoitteet:

- http://wilma.kunta.fi/teachers/1/1
- http://wilma.kunta.fi/teachers/1/2
- http://wilma.kunta.fi/teachers/2/2
- http://wilma.kunta.fi/teachers/2/1

Hakasulkeissa olevien palojen maksimimäärä rajattiin neljään, mutta tälläkin määrällä voidaan yhdestä www-osoitteesta pakottaa Wilma URL Pommittaja rakentamaan tuhansia www-osoitteita.

Itse www-osoitteiden käsittely oli monimutkainen osa ohjelman koodia. Koodissa tuli silmukkaa toisen sisään ja näiden lisäksi täytyi vielä huomioida esimerkiksi www-osoitteen oikein rakentaminen. Loppupeleissä Wilma URL Pommittajaan kuluneesta ajasta noin kolmasosa meni juurikin tämän osa-alueen suunnitteluun, ohjelmointiin ja testaukseen.

5.5 Pommituksen etenemisen näyttäminen käyttäjälle

Pommitustapahtuma voi parhaimmillaan kestää muutamia minuutteja, joten sovelluksen on tärkeää antaa reaaliaikaista informaatiota käyttäjälle pommituksen etenemisestä.

Eteneminen näytetään käyttäjälle sovelluksen pääikkunan alaosassa kahdella tavalla:

- Reaaliaikaisena numeraalisena arvona, kuinka monta osoitetta on käyty läpi.

- Valmistumispalkkina, joka kuvaa graafisesti pommituksen edistymisen.

Tilanne, jossa pääikkunaa päivitetään jatkuvasti läpikäydyillä tiedoilla, on sovellukselle raskas. Päivittäminen aiheutti aluksi tilanteita, joissa koko sovellus jähmettyi kunnes pommitustapahtuma oli käyty kokonaan loppuun. Tilanne päädyttiin ratkaisemaan siten, että ajetaan kaikki sovelluksen raskaat tehtävät, eli tässä tapauksessa Wilman osoitteiden rakentaminen ja itse pommitus erillisessä säikeessä (*Thread*). Tällöin eri säie päivittää sovelluksen pääikkunaa eikä raskas pommitus pääse jähmettämään käyttäjälle näkyvää tapahtumaa omalla raskaudellaan.

```
procedure TUFormMain.UpdateStatusBar(value:integer);
const
   cMessage = 'Ajettu %d osoitetta.';
begin
   ProgressBar.Position:=value;
   ProgressBar.Refresh;
   StatusBar.Panels[0].Text:=Format(cMessage, [value]);
   StatusBar.Refresh;
end;
```

Kuva 5.2. Koodi, jota käyttäen erillinen säie päivittää käyttäjän näkemän pääikkunan.

5.6 Pommitustapahtuma

Kun sovellus on prosessoinut www-osoitteista lopullisen listan, aloittaa se itse pommitustapahtuman. Käyttäen hyväksi Synapse-kirjaston mukana tulleita funktioita, sovellus lähettää jokaiseen kohde Wilman www-osoitteeseen yksi kerrallaan kyselyn. Wilma vastaa jokaiseen kyselyyn ilmoituksella kyselyn onnistumisesta sekä virhekoodilla.

HTTP Virhekoodi	***Mitä kyseinen virhekoodi tarkoittaa***
200	Käyttäjällä on oikeus nähdä sivun sisältö.

403	Käyttäjällä ei ole oikeutta nähdä sivun sisältöä.
404	Kyseistä sivua ei löytynyt.
500	Ajetun toiminnon toteuttaminen aiheutti virheen ohjelman lähdekoodissa.

Taulukko 5.1. HTTP Virhekoodit ja niiden selitykset (Google Inc, 2014).

Mikäli Wilma palauttaa sovellukselle virhekoodin 200 (taulukko yllä), sovelluksella on oikeus nähdä Wilma palvelimen palauttama sivu. Tässä tilanteessa sovellus kirjaa tapahtuman hyväksytyksi ja siirtyy seuraavaan osoitteeseen listalla (Google Inc, 2014).

Jos palautuva virhekoodi on muotoa *4XX* tai *5XX* alkava, lataa sovellus Wilman palauttaman www-osoitteen HTML-koodin. Sovellus käy läpi tämän HTML-koodin ja etsii siitä Pascaliin kuuluvalla *Pos*-funktiolla *<title></title>* -HTML-tagien sisältämän selvennyksen virhekoodille. Tämä haettu selvennys lisätään tulosten näyttämistä varten tallennettavaan tietueeseen ja jatketaan seuraavaan listalta löytyvään osoitteeseen (Google Inc, 2014).

Sovellus kokoaa listaa jokaisesta www-osoitteesta ja sen palauttamasta tuloksesta. Kun Wilman pommitus on saatu päätökseen, kasataan nämä tulokset yhteen ja näytetään käyttäjällä erikseen avattavassa tulosikkunassa.

5.7 Tulosten näyttäminen käyttäjälle

Tulokset näytetään käyttäjälle erillisessä tulosikkunassa sijaitsevassa taulukossa, joka muistuttaa hyvin paljon esimerkiksi taulukkolaskennassa käytettyä taulukkopohjaa. Sarakkeet on värjätty *Tila*-sarakkeen mukaan joko *punaiseksi* tai *vihreäksi*. Rivit värjätään eri väreillä siksi, että käyttäjän on mahdollista nopeasti löytää ja erottaa virherivit täsmäävien rivien joukosta.

Käyttäjällä on tässä ikkunassa mahdillista myös raja tuloksia kahdella rastikentällä taulukon alalaidassa:

- *Vain virheet* -rastikenttää rastitettaessa sovellus poistaa tulostaulukosta kaikki ne rivit, joiden *Tila*-sarakkeessa on jokin muu tieto kuin ”VIRHE”.

- *Älä näytä 404 palauttaneita* -rastikenttä poistaa taulukosta kaikki ne rivit, joiden palautuskoodi sarakkeessa on tieto 404. Tämä suodatin on mukana sen vuoksi, että saadaan ns turhat, tietoa sisältämättömät sivut pois sotkemasta näytettäviä tuloksia.

Yllä mainittuja suodattimia voidaan myös käyttää yhteen, jolloin taulukossa näytetään vain virheelliset rivit, joiden palautuskoodi on eri kuin 404. Tämä on yleensä se tulos, jota käyttäjä tältä ohjelmalta haluaa.

Tällainen taulukkolaskennasta tuttu pohja voi käydä tilanteesta riippuen sovellukselle raskaaksi, koska käsiteltäviä rivejä saattaa olla jopa tuhansia. Taulukon nopeutta saatiin parannettua taulukkopohjan mukana tulevilla *BeginUpdate*- ja *EndUpdate*-proseduureilla, jotka estävät sovellusta näyttämästä taulukon päivitystä reaaliaikaisesti käyttäjälle, vaan hänelle näytetään taulukossa tapahtuneet muutokset vasta, kun ne on lopullisesti tehty. Näin sovelluksen ei tarvitse tuhlata tietokoneen resursseja päivittääkseen käyttäjälle näkymää esimerkiksi jokaisen rivin tallentamisen jälkeen.

Tulosikkunasta käyttäjä voi myös tallentaa tuloksista tekstitiedoston, jota hän voi myöhemmin käyttää mukana vertailussa. Tähän tiedostoon tallennettaan aina kaikki tulosrivit, joten yllä mainitut suodattimet eivät vaikuta tiedoston rakentamiseen.

5.8 Virhetilanteiden hallinta

Sovelluksen lähdekoodissa on pyritty mahdollisimman hyvin huomioimaan erilaiset virhetilanteet, joita käyttäjän on mahdollista saada aikaiseksi. Normaalisti osa tällaisista virhetilanteista aiheuttaisi koko sovelluksen sammumista, mutta jos kyseinen virhetilanne on osattu ennakoida jo sovelluksen lähdekoodia kirjoitettaessa, voidaan virhe kuitata esimerkiksi ilmoituslaatikkona käyttäjälle.

Virhetilanteiden hallintaa tulee sovelluksessa muun muassa vastaan, kun yritetään aloittaa pommitustapahtumaa. Tällöin sovellus ottaa yhteyden kohteena olevaan Wilmaan, mutta jos kyseinen Wilma ei vastaakaan, niin käyttäjälle ilmoitetaan ponnahdusikkunana Wilmaan yhteydenoton epäonnistumisesta ja sovelluksen käyttöä voi tämän jälkeen jatkaa normaalisti.

Toinen yleinen tilanne, joka aiheuttaa tällaisia virhetilanteita on sovelluksen koodissa tehtävät tyyppimuunnokset. Jos muutetaan esimerkiksi merkkijono -tyyppistä tietoa (*string*) numeraaliseksi muuttujaksi (*integer, cardinal, int64*), mutta muutettava tieto ei olekaan numeraalinen tieto, ohjelman suorittaminen päättyy ja käyttäjälle ilmoitetaan EConvertError -virheilmoitus.

Sovelluksen *asetukset.txt* -tiedostosta on myös mahdollista laittaa päälle Debug -toiminto, joka kirjaa sovelluksen tapahtumat ja muuta ehkä tulevaisuudessa tarpeellista tietoa *debug.txt* -nimiseen tekstitiedostoon. Tätä toimintoa voidaan käyttää virheen paikantamiseen, jos sellaisia myöhemmässä käytössä ilmenee. Suurin osa virheistä toivottavasti tuli esille jo sovelluksen toteuttamis- ja testaamisvaiheissa.

5.9 Sovelluksen tulevaisuus

Wilma URL Pommittajaa tullaan varmasti kehittämään tulevaisuudessa sopimaan paremmin sille annettaviin tehtäviin. Ensimmäinen suurempi muutos koskettaa www-osoitteiden käsittelyä ja tuon numeroalueen [1-5] lisäksi olisi tarkoitus tehdä myös kirjainalueita, joissa käyttäjä voisi samaan tapaan määritellä esimerkiksi [a-k] -osion www-osoitteeseen mukaan. Tämä itsessään luo jo omanlaisensa haasteen, koska kirjaimien käsittely on mielestäni tässä tapauksessa vaikeampaa kuin numeroiden käsitteleminen.

Sovellusta ollaan vasta tämän opinnäytetyön kirjoittamisen aikaan ottamassa käyttöön, joten tarkempia kehityskohtia tulee vastaan, kun ohjelman käyttäminen tuotannossa on aloitettu. Käyttöönotto vaatii ison työn demossa käytettävän tietokannan tekemiseksi Primus-ohjelmaan.

Näkisin itse mielelläni ohjelman tulevaisuudessa kehittyvän niinkin laajaksi, että siitä olisi hyödyllistä tehdä erikseen palvelin ja asiakasohjelmisto –osat, tällöin voitaisiin ottaa mukaan myös MySQL-tietokanta. Palvelin – asiakasohjelmisto -suhde mahdollistaisi monen yhtäaikaisen käyttäjän käyttää samaan aikaan samoja profiileja ja www-osoitteita.

6 JOHTOPÄÄTÖKSET

Omasta mielestäni lopullinen sovellus vastasi sitä, mitä siltä alun perin vaadittiinkin, joten opinnäytetyön tavoitteisiin päästiin. Sovellus pystyy oikein käytettynä huomaamaan mahdolliset ongelmat käyttöoikeuksissa ja se on helppokäyttöinen. Parantamisen varaa sovellukseen jäi muun muassa käyttöliittymän suunnittelussa, joka olisi voinut olla hiukan selkeämpi ja helpommin käytettävissä. Sovellusta käyttävät sovellustestaajat varmasti saavat sovelluksesta sen hyödyn, mitä varten se on rakennettu alun alkaen.

Wilma URL Pommittajaa ei vielä ole otettu täysipäiväiseksi työkaluksi sovellustestaamiseen StarSoft Oy:llä, mutta alkuvalmistelut sen käyttöön ottoon on aloitettu. Sovelluksen käyttöönotto vaatii pohjalle hyvin rakennetun realistisen tietokannan StarSoft Oy:n ohjelmille.

6.1 Oma oppiminen

Alun perin pidin tätä sovellusprojektia vähän pelottavana, koska projektin idea oli mielestäni haastava kokonaisuus. Pääsin omasta mielestäni kuitenkin nopeasti perille siitä, mitä sovellukselta toivottiin ja millaisia ongelmatilanteita sen tulisi käytössä ollessaan ratkaista.

Omasin jo alun perin perusalkeet Pascal-ohjelmointikielestä, mutta projektin avulla opin paljon järkeistämään omaa tuottamaani koodia. Tällä sain koodista helpommin ymmärrettävää, siistimpää sekä myöhemmin helpommin muokattavaa. Opin myös olemaan paljon pitkäjänteisempi ja tutkimaan asioihin kunnollista ratkaisua. Kunnon ratkaisu on kuitenkin aina parempi, kun mennä siitä mistä aita on matalin.

Suurena asiana tahtoisin ottaa esille Ararat Synapse -kirjaston tuomien verkkoratkaisujen käyttämisen, joka oli todella mielenkiintoista, ja opin sen avulla paljon uutta verkkoliikennettä käyttävien sovellusten rakenteesta ja toimintaperiaatteista. Tätä kirjastoa on tullut käytyä läpi Wilma URL Pommittajan tekemisen jälkeen-

kin, ja voin suositella sen käyttöä kaikille, jotka Pascal-ohjelmointikieltä käyttävät ja tarvitsevat kyseisiä toimintoja.

Projekti myös opetti minulle sen, että en vielä ole täysin valmis sovellusohjelmoijaksi. Paljon asioita on vielä pimennossa, mutta omasta mielestäni olen hyvää vauhtia menossa kohti tavoitettani työskennellä ohjelmoinnin saralla. Wilma URL Pommittajan tekeminen antoi myös takaisin sen ehkä hiukan kadonneena olleen kipinän, jonka avulla jaksan myöskin vapaa-aikanani taas harrastaa ohjelmointia, ja tästä asiasta olen erittäin iloinen.

LÄHTEET

Ararat Synapse 2007. Project Idea of Synapse. Viitattu 15.02.2014. Saatavilla www-muodossa: http://synapse.ararat.cz/doku.php/about

Google Inc 2014. Webkehittäjän työkalut, HTTP-tilakoodit. Viitattu 13.02.2014. Saatavilla www-muodossa: https://support.google.com/webmasters/answer/40132?hl=fi

Lazarus 2013. About Lazarus Project. Viitattu 13.02.2014. Saatavilla www-muodossa: http://www.lazarus.freepascal.org/index.php?page=about

Software Testing Fundamentals 2014. Test plan. Viitattu 13.03.2014. Saatavilla www-muodossa: http://softwaretestingfundamentals.com/test-plan/

Software Testing Help 2014. How to Report Test Execution Smartly. Viitattu 13.03.2014. Saatavilla www-muodossa: http://www.softwaretestinghelp.com/test-execution-report/

StarSoft Oy 2012. Referenssi. Viitattu 18.02.2014. Saatavilla www-muodossa: http://www.starsoft.fi/public/?q=node/10

StarSoft Oy 2012. Tuote-esittely. Viitattu 18.02.2014. Saatavilla www-muodossa: http://www.starsoft.fi/public/?q=node/7

StarSoft Oy 2014. Sisäinen testausdokumentti. Viitattu 24.02.2014.

StarSoft Oy 2014. Sisäinen suunnitteludokumentti. Viitattu 24.02.2014.

Techopedia, 2014. Use case. Viitattu 13.03.2014. Saatavilla www-muodossa: http://www.techopedia.com/definition/25813/use-case

Tietoviikko 2011. VR myöntää: it-ongelmat olisi pitänyt tunnistaa etukäteen. Viitattu 12.03.2014. Saatavilla www-muodossa: http://www.tietoviikko.fi/cio/vr+myontaa+itongelmat+olisi+pitanyt+tunnistaa+etukateen/a697637

Tutorialspoint 2014. Software Testing Methods. Viitattu 13.03.2014. Saatavilla www-muodossa: http://www.tutorialspoint.com/software_testing/testing_methods.htm

Webopedia. Pascal. Viitattu 11.03.2014 Saatavilla www-muodossa: http://www.webopedia.com/TERM/P/Pascal.html