LAUREA

Kytkentöjen määrän visualisoiminen Javalla DSiP-lokitiedostoista

• •

Bertling, Mikko

2011 Leppävaara

Laurea-ammattikorkeakoulu
Laurea Leppävaara

Kytkentöjen määrän visualisoiminen Javalla DSiP-lokitiedostoista

Bertling, Mikko
Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
Maaliskuu, 2011

Laurea-ammattikorkeakoulu
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Bertling, Mikko

Kytkentöjen määrän visualisoiminen Javalla DSiP-lokitiedostoista

Vuosi 2011 Sivumäärä 25

Luotettavien järjestelmien valmistamiseen erikoistunut Ajeco Oy on kehittänyt DSiP-monikanavareititysjärjestelmän ja tarvitsi sille lokitiedostojen analysointiohjelman helpottamaan verkon hallintaa. Tavoitteena oli kehittää tutkimustieteellisen lähestymistavan mukaisesti laitteistoriippumaton ohjelma, joka mahdollistaa lyhyellä ja pitkällä aikavälillä verkon tilan tarkistelemisen visuaalisesti. Ohjelma kirjoitettiin Javalla, koska se on laitteistoriippumaton ohjelmointikieli.

Annettujen rajojen sisälle laadittiin arkkitehtuuri, jonka pohjalta toteutettiin vaatimukset täyttävä ohjelma. Kehitystyön edistyessä tuli ilmi myös tarve yleisnäkymälle. Valmiina olevien osien pohjalta johdettiin yleisnäkymä, joka mahdollistaa koko verkon tilan tarkistelemisen raja-arvojen perusteella.

Ohjelma toteuttaa hyvin sille asetetut tavoitteet. Työ toteutettiin itsenäisesti ja kehityskommentteja huomioon ottaen. Ohjelmointityö ja graafinen käyttöliittymä toteutettiin hyvin yleisen vaatimustason mukaisesti. ”Ohjelma piirtää DSiP-järjestelmän lokitiedostojen pohjalta ymmärrettävän graafin, joka auttaa käyttäjää näkemään noden kytkeytymistiheyden. Katsomme osaltamme, että tehty työ on ONT:n kannalta kiitettävä.” (Holmström 2011.) Työ osoitti, että Java soveltuu erittäin hyvin ammattimaiseen laitteistoriippumattomaan koodaukseen, jossa lopputuotos toteuttaa vakioelementtejä laajentavan graafisen käyttöliittymän.

Asiasanat DSiP, Java, visualisoiminen

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Programme in Business Information Technology

Bertling, Mikko

Visualization of connection counts from DSiP log files with Java

Year 2011 Pages 25

Ajeco Oy has developed a DSiP (Distributed Systems intercommunication Protocol) multichannel solution to provide reliable networking for critical applications such as law enforcement. They needed a system independent tool to help them analyze the state of the network on a short and long term scale. The objective was to code a program that would accomplish this through analysing the DSiP log files. Java was chosen as the coding language since it is a system independent language.

Within the given constraints a software architecture was drawn up and based on that the program was developed. During the development process it became apparent that an over arching view was needed as well. Thus the program was further developed based on this feedback and an overview tab was added to it. This enables the program to provide the big picture of the entire network at one glance.

The project achieves the objectives set for it. The code and graphical user interface are up to expectations. The project was executed independently and development proposals were taken in to account. "Based on the log files, the program draws an understandable graph that helps the user to see how often the node has connected and disconnected. From our point of view the project has been completed commendably." (Holmström 2011.) This project demonstrated that Java is a good choice for developing a platform independent program that extends the standard graphical user interface.

Key words DSiP, Java, visualization

Sisällys

1 Johdanto 6
1.1 Taustaa 6
1.2 Tavoitteet 6
2 Käsitteet 7
2.1 DSiP 7
2.2 Java 7
2.3 Tutkimustieteellinen lähestymistapa 8
3 Arkkitehtuuri 8
3.1 Runko 9
3.2 Graafit 11
4 Testaus 13
4.1 Menetelmä 14
4.2 Havainnot 14
4.3 Korjaukset 15
5 Iterointi-vaihe ja lopputulos 16
6 Jatkokehitysehdotukset 16
7 Arviointi 18
8 Yhteenveto ja johtopäätökset 18
Lähteet 20
Liitteet 22

1 Johdanto

Ajeco Oy on kehittänyt DSiP-monikanavareititysjärjestelmän, jolla viranomaiset sekä muut kriittisten yhteyksien päässä olevat tahot pystyvät toteuttamaan luotettavan ja vikasietoisen IP-ympäristön. Kyseinen järjestelmä kerää lokeja tapahtumista, mutta niiden läpikäymiseen ei ollut sovelluksia. Tämä tarkoittaa sitä, että järjestelmävastaavat joutuivat käymään tekstimuotoisia tiedostoja lävitse hahmottaakseen verkon tapahtumat. Tässä opinnäytetyössä luotiin ohjelma, joka visualisoi kytkentöjen määrän muuttumisen järjestelmässä lokitietojen perusteella, jotta järjestelmän tilan pystyisi hahmottamaan havainnollisemmin ja nopeammin. Idea ohjelman toteutukselta tuli Ajeco Oy:ltä.

1.1 Taustaa

DSiP-järjestelmän on kehittänyt vuonna 1984 perustettu Ajeco Oy, joka on erikoistunut luotettavien järjestelmien valmistamiseen. DSiP-järjestelmän patentointiprosessi oli vielä hankkeen alkaessa kesken, joten hanke nojautuu vahvasti Ajeco Oy:n asiantuntijoilta saatuun julkaisemattomaan tietoon.

”Eräs DSiP-järjestelmän perusfilosofia on, että node-ohjelma pyrkii säilyttämään yhteyden DSiP-reitittimeen ollen "on-line" jatkuvasti. Erilaisista syistä johtuen node saattaa kuitenkin poistua reitityksestä ja uudelleen kytkeytyä toistuvasti. Esimerkiksi huono 3G yhteys saattaa johtaa yhteyksien pätkintään. Pätkiminen saattaa indikoida esimerkiksi huonoa 3G kuuluvuutta tai peräti ongelmia antenneissa jne. Kytkeytymistiheyden näyttäminen graafina auttaa järjestelmän ylläpitäjää havaitsemaan ongelmalliset nodet.” (Holmström 2011.)

Tämän opinnäytetyön lisäksi myös Sami Alaverronen toteuttaa lokitiedostojen visualisoimista DSiP-ympäristöön, mutta hän keskittyy liikennemäärien visualisoimiseen. Hankkeen käyttämät lokitiedostot ovat peräisin Laurean Tietoverkkopalvelujen kehittäminen -kurssilla toteutetusta DSiP-ympäristöstä.

1.2 Tavoitteet

Opinnäytetyön tavoitteena oli kirjoittaa Java-ohjelma, jonka on tarkoitus helpottaa DSiP-monikanavareititysjärjestelmän hallintaa. Ohjelman oli tarkoitus seurata lokitiedostojen perusteella verkon kytkentöjen määrää päivä-, viikko- ja kuukausitasolla. Ohjelma pyrittiin kehittämään laitteistoriippumattomaksi ja toteuttamaan toimeksiantajan pyynnöstä englanninkielisenä.

2 Käsitteet

Tässä esitellyt termit kattavat opinnäytetyön kannalta oleellisimmat käsitteet. Opinnäytetyön osana syntynyt ohjelma ei varsinaisesti ole suoraan vuorovaikutuksessa DSiP-verkon toiminnan kanssa, joten lukijalta ei vaadita tämän syvempää ymmärrystä sen osalta.

2.1 DSiP

DSiP-järjestelmä (Distributed Systems intercommunication Protocol) on ympäristö, joka tarjoaa turvattua ja luotettavaa tiedonsiirtoa monikanavareititysympäristössä. Se soveltuu erityisesti käytettäväksi ympäristöissä, joissa tiedonsiirto on kriittistä, kuten esimerkiksi sähköverkon kaukohallinnassa.(Holmström 2010). DSiP pohjautuu IP-protokollaan lisäten siihen ominaisuuksia jotka tekevät siitä luotettavamman ja turvallisemman.

Monikanavareititysympäristö tarkoittaa, että yhteyttä pitävien kohteiden on mahdollista ylläpitää yhteyttä monen erilaisen yhteyskäytännön tai tekniikan kautta, esimerkiksi kiinteän ASDL-linjan ja 3G-verkon kautta. DSiP-järjestelmässä on myös mahdollista ylläpitää samaan aikaan yhteyttä monen tekniikan yhteistyöllä, eli näin pystytään myös käyttämään suurempaa tiedonsiirtokapasiteettia verkossa.

DSiP-järjestelmässä on kahdenlaisia toimijoita. DSiP-palvelimet toimivat verkon selkärankana ja liikenteen risteyspisteinä. Niillä on myös mahdollista toteuttaa verkon hallintaa ja seurantaa. Tässä hankkeessa käytettiin network management -palvelimen keräämiä lokitiedostoja yhteysmäärien selvittämiseen. ”Lokitiedostot sisältävät lauseita, joista käy ilmi ajankohta milloin node on liittynyt ja poistunut järjestelmästä.” Verkon laidoilla toimii DSiP-nodeja; nämä ovat käytännössä päätelaitteita kuten esimeriksi turvakamerat tai niitä hallinnoivat tietokoneet. (Holmström 2010; Holmström 2011.)

2.2 Java

Java-ohjelmointikieli on ilmainen oliopohjainen alustariippumaton monisäikeinen ohjelmointikieli ja alusta. Javan alustariippumattomuus mahdollistaa sen, että ohjelma kirjoitetaan yhdellä kielellä ja sen jälkeen vain käännetään Java-tulkilla kohdealustalle. Näin ohjelmiston kehittäjä ei joudu kirjoittamaan koodia uusiksi jokaista käyttöjärjestelmää varten. Javan ja sen Netbeans-kehitysympäristön ilmaisuus teki siitä erittäin soveltuvan tämän hankkeen toteuttamiseen. Javan kehitti alun perin Sun Microsystems vuonna 1995. Oracle osti Sun Microsystemsin, joten Java siirtyi sen alaisuuteen (Oracle).

2.3 Tutkimustieteellinen lähestymistapa

Tutkimuksellisena lähestymistapana käytettiin tutkimustieteellistä lähestymistapaa ja se määritti hankkeen toteutuksen suuntaviivat. Lähestymistapa sisältää Hevnerin, Marchin, Parkin ja Ramin mukaan seitsemän kohtaa (2004, 82-90):

(1.) Tutkimustieteellisen lähestymistavan mukaisesti tarkoituksena on toteuttaa toimiva tietotekninen luomus.

(2.) Tavoitteena on tuottaa teknologialähtöinen ratkaisu tarpeelliseen ja ajankohtaiseen yritysmaailman ongelmaan.

(3.) Luomuksen käyttökelpoisuus, laatu ja vaikutus täytyy näyttää toteen.

(4.) Tehokas suunnittelutieteellinen tutkimus pyrkii saavuttamaan selkeitä ja todistettavissa olevia saavutuksia luomukseen, suunnittelun perusteisiin tai suunnittelumenetelmiin.

(5.) Suunnittelutieteellinen lähestymistapa rakentuu täsmällisten metodien käyttämiseen niin luomuksen kehittämisessä kuin arvioimisessa.

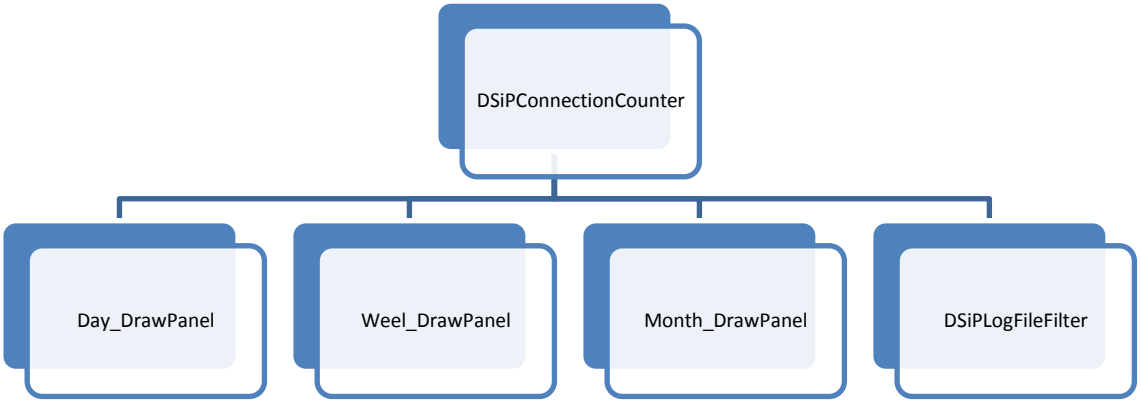
(6.) ”Suunnittelu etsintäprosessina” tarkoittaa sitä, että tutkimuksen pitää käyttää käytettävissä olevia keinoja saavuttaa tavoiteltu lopputulos, mutta samalla pysyä ongelma alueen rajojen sisäpuolella. Suunnittelutieteellinen lähestymistapa on luonnostaan iteratiivinen, joten parhaan lähestymistavan löytäminen on käytännössä etsintäprosessin lopputulos.

(7.) Suunnittelutieteelliset tutkimustulokset täytyy esittää tehokkaasti niin tekniikkaan kuin liiketoiminnan hallintaan suuntautuneelle yleisölle.

3 Arkkitehtuuri

DSiP-monikanavareititysympäristö voi sisältää määrittelemättömän määrän DSiP-nodeja sekä palvelimia eikä niiden lokitiedostojen koolle ole asetettu mitään rajaa, joten ohjelman arkkitehtuuri on suunniteltu tämän lähtökohdan ympärille. Arkkitehtuuriratkaisuilla on siis pyritty varautumaan siihen, että jossain vaiheessa tiedostojen määrät tai koot saattavat kasvaa niin isoiksi, että ne häiritsisivät ohjelman toimintaa. Tämä tarkoittaa käytännössä sitä, että esimerkiksi tiedoston lukemisessa pyrittiin sisällyttämään kaikki logiikka yhden toiminnon taakse, sen sijaan että sitä olisi hajautettu eri funktioiden ja tiedostorakenteiden sekaan.

Ohjelma koostuu viidestä luokasta. Ohjelman rakenne näkyy kuvasta 1 niin, että runkona toimivan DSiPConnectionCounterin alla toimii 3 graafielementtiä ja tiedostojen suodattamiseen käytetty luokka. Tästä johtuen esimerkiksi luettua dataa ei sijoitettu välissä omaan tiedostorakenteeseensa, vaan se välitettiin suoraan graafielementille. Myös ohjelman liikuttelemien tietoelementtien koko on pyritty valikoimaan mahdollisimman pieneksi, jotta ohjelma ei hidastu liikaa tiedostojen kokojen kasvaessa.

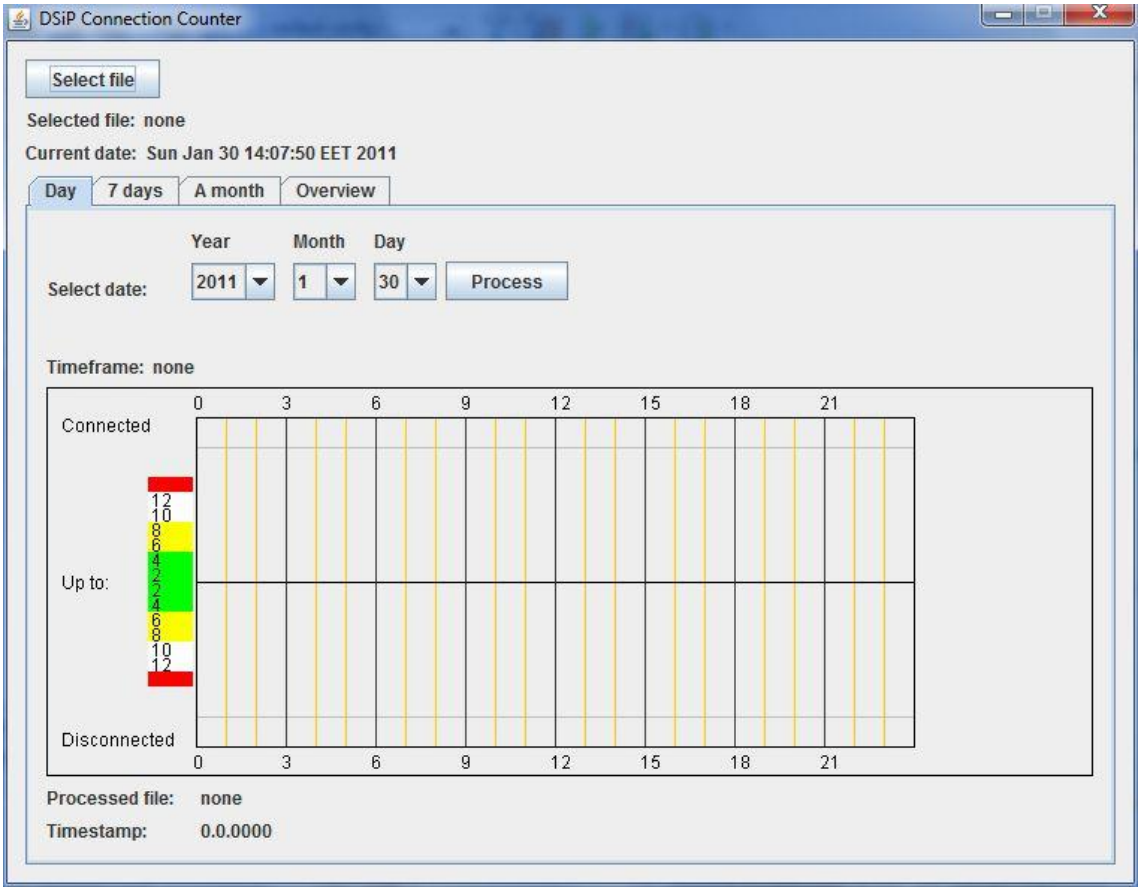


Kuva 1: Ohjelman rakenne

Ohjelman kirjoittamisessa otettiin myös huomioon se, että sitä saatetaan tahtoa jossain vaiheessa muokata hyvinkin rajoittuneille kohdealustoille. Tämän takia pisteiden sijoittelu aikajanalle on toteutettu case-rakenteella, jotta tarvittaessa ohjelman muokkaaja voi helposti määritellä aikavälit epätasaisin välein. Ohjelma toteutettiin Javan omilla ominaisuuksilla. Näin se pystyy toteuttamaan vaatimuksina olleen laitteistoriippumattomuuden (Holmström & Ramstedt 2010) sekä tarjoamaan vakaamman rungon koko ohjelmalle.

3.1 Runko

DSiPConnectionCounter-luokka huolehtii ohjelman käyttöliittymästä ja yleisestä toimintalogiikasta. Sen kautta hoidetaan niin tiedostojen paikantaminen kuin niiden läpi käyminen. Kuten kuvasta 2 näkyy, DSiPConnectionCounter sisältää käyttöliittymän tiedoston valitsemiseen ja sen ala puolella on välilehdille jaettuna päivä-, viikko- ja kuukausigraafien tila, sekä Overview-välilehti. Ohjelman jokainen graafi on toteutettu omana luokkanaan ja omana välilehtenään, jotta niitä voi tarkastella riippumatta toisen tilasta.



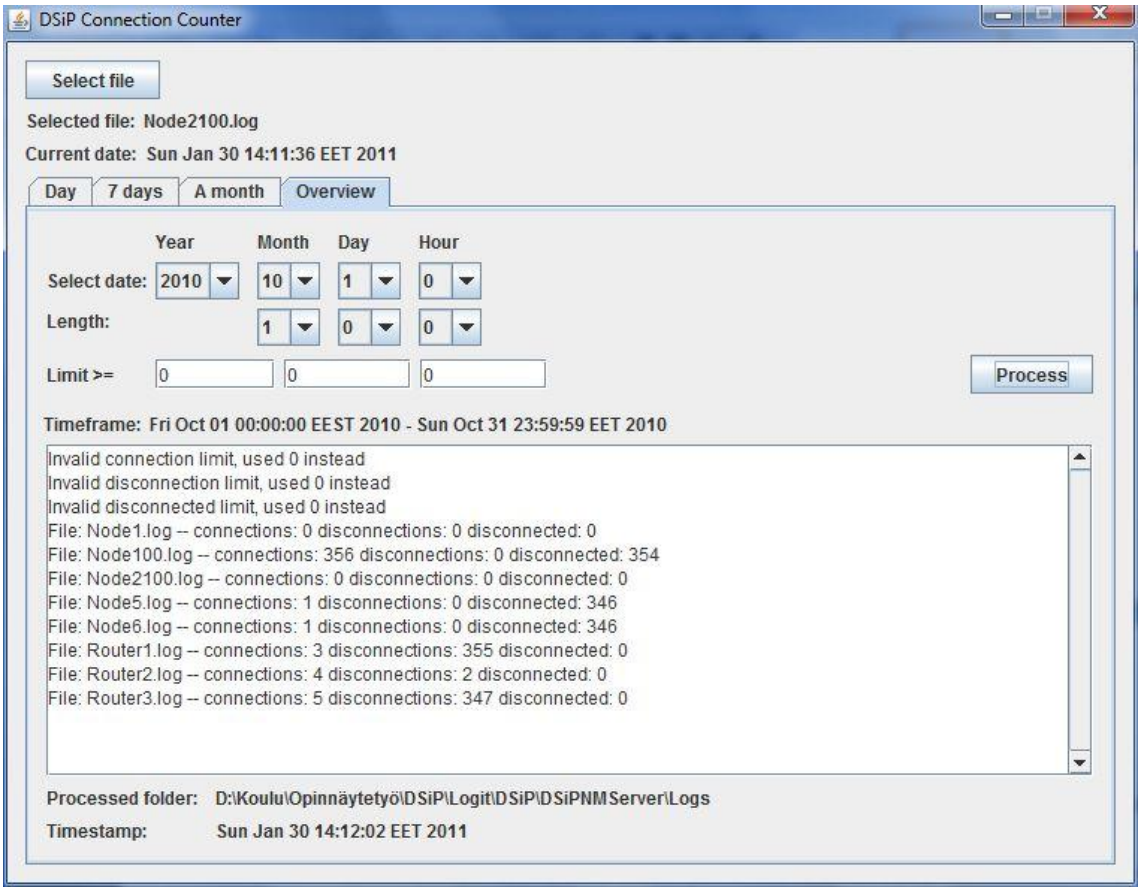
Kuva 2: Aloitusnäkymä

Rungon koodi koostuu lähinnä käyttöliittymän toimintalogiikan ja jokaisen välilehden tiedostonlukulogiikan toteuttavasta koodista. Runko huolehtii tiedon sisään lukemisen aikana oikeiden tietojen syöttämisestä graafeille. Kun kaikki yksityiskohtaiset tiedot ja yhteenvetotiedot on kerätty kyseisestä tiedostosta, pyytää se graafia piirtämään kyseisen esityksen näkyville.

Tämän lisäksi runko huolehtii yhdessä graafielementtien kanssa selventävien tekstien piirtämisestä graafien päälle, kuten kuvasta 5 tulee ilmi. Graafin piirtämiseen liittyvä logiikka on pyritty pitämään graafiluokkien puolella, tai välitetty sen osat parametreina rungolle. Sen sijaan Overview-välilehden kaikki toiminnallisuus on toteutettu rungon sisällä, sillä se ei piirrä mitään.

Overview-välilehdellä käyttäjä voi valita valittuna olevan tiedoston hakemistosta kaikkien nodejen ja palvelimien tiedot kerralla käsiteltäviksi. Tämän jälkeen määritellään haluttu aikaväli ja rajat tapahtumille. Näiden perusteella ohjelma käy läpi kaikki hakemistossa olevat tiedostot ja kirjoittaa niistä tekstikenttään omalle rivilleen tiedot yhteyksistä ja yhteyksien katkeamisista, jos ne ylittävät raja-arvot. Kuva 3 havainnollistaa Overview-välilehden tuottamaa tekstiä, sekä sen kuinka ohjelma ilmoittaa käyttäjälle jos hän yrittää syöttää

epäkelpoja raja-arvoja. Tämä näkymä mahdollistaa isoistakin verkoista ongelmakohtien paikantamisen nopeasti tarkempaa tarkistelua varten.



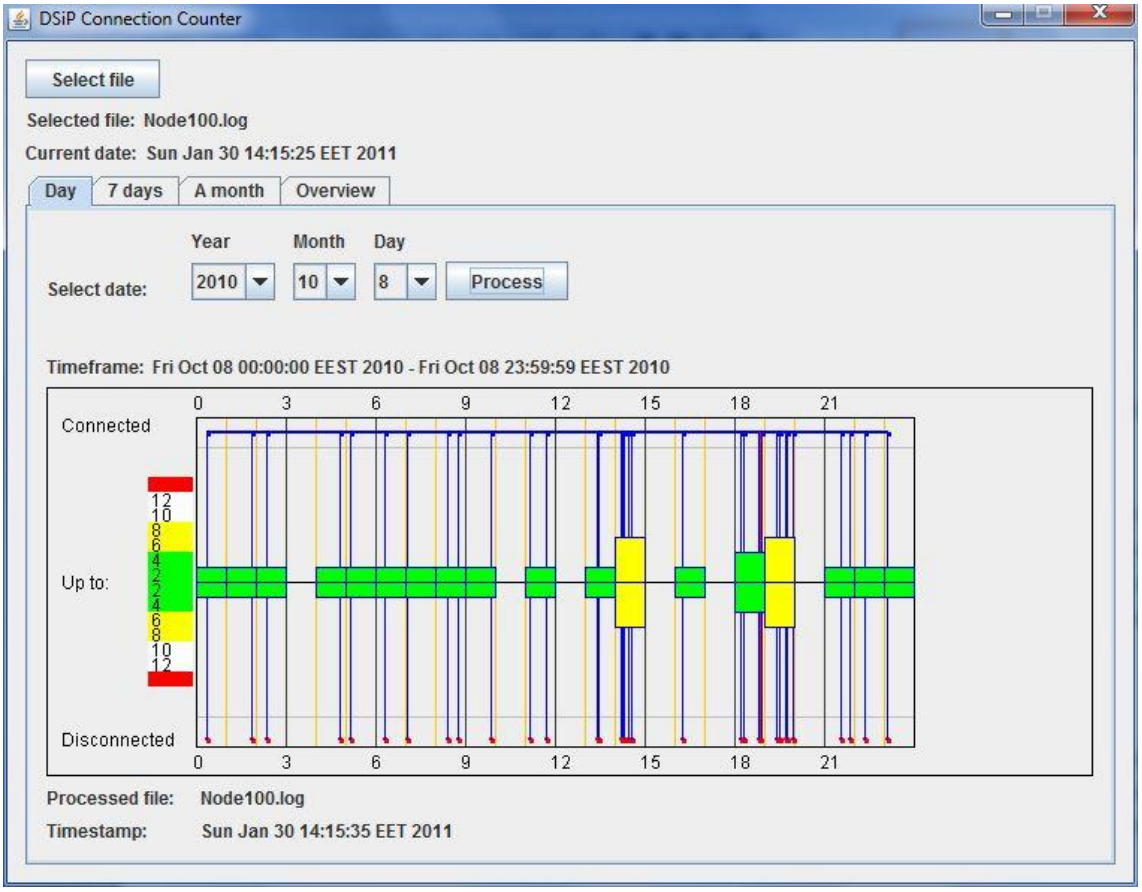
Kuva 3: Overview-välilehti

DSiPConnectionCounter-luokka on myös ainoa luokka hankkeessa joka käyttää hyväkseen DSiPLogFileFilter-luokkaa. Tämän avustavan luokan kautta mahdollistetaan se, että käyttäjän on mahdollista valita ohjelman käsiteltäväksi vain DSiP-lokitiedostoja. Luokkaa käytetään tiedostojen valikoimiseen käytettävän valikkoikkunan logiikan yhteydessä. Valitettavasti avustavaa luokkaa ei voitu käyttää Overview-välilehdellä toimintalogiikkaa avustamaan. Vaikka dokumentaatio väitti sen olevan sopiva luokka, käytettävät metodit eivät hyväksyneet sen käyttöä ja toimintalogiikka jouduttiin kirjoittamaan sinne erikseen.

3.2 Graafit

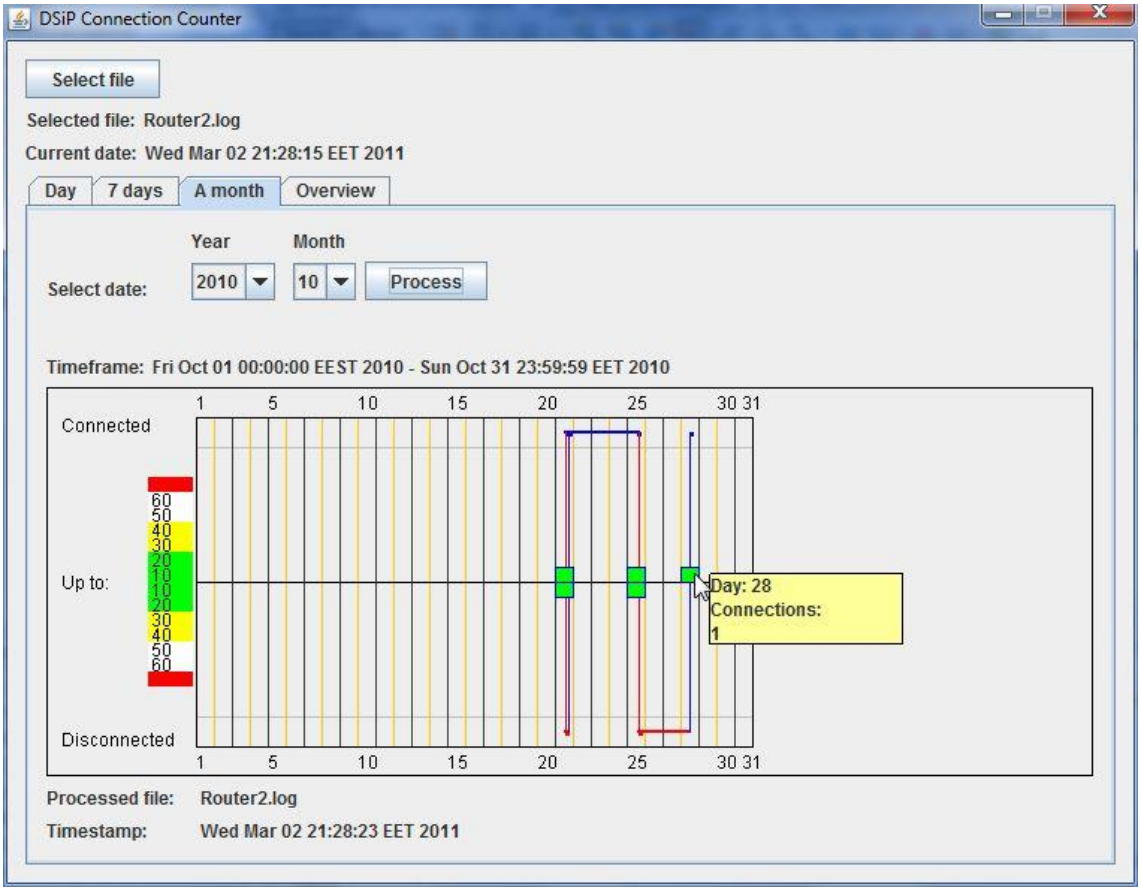
Graafiluokat piirtävät graafien taustaruudukon, yhteyspisteet sekä -viivat ja yhteenvetopalkit. Yhteyspisteet piirretään aikajanalle vasemmalta oikealle ylä- ja alareunaan sen perusteella, ovatko ne verkkoon yhteydessä vai ilman yhteyttä, kuten kuvista 4 ja 5 voi nähdä. Yhteyden tilaa kuvaavat pisteet määritellään suoraan lokitiedostoista saatujen tietojen perusteella. Yhteyttä kuvaavat pisteet ovat sinisellä ja yhteydetöntä tilaa kuvaavat

pisteet punaisella. Pisteiden väliin luokat piirtävät viivoja jotka kuvaavat oliko node tai serveri näiden pisteiden välisenä aikana yhteydessä vai ilman yhteyttä verkkoon. Esimerkiksi kahden yhteyden puuttumista kuvaavan pisteen välille ohjelma piirtää punaisen viivan. Jos seuraava piste on yhteyttä kuvaava piste, ohjelma piirtää punaista viivaa ruudun alalaidassa siihen kohtaan asti aikajanalla jossa yhteyttä kuvaava piste on. Tämän jälkeen se piirtää sinisen viivan siitä kohdasta ylöspäin yhteyttä kuvaavaan pisteeseen, näin yhdistäen pisteet viivalla kuten kuvassa 5. Pisteiden väliseen tilaan piirretään yhteenvetopalkit, jotka antavat nopeallakin silmäyksellä kuvan siitä, kuinka paljon yhteyksiä tai katkoksia aikavälillä on tapahtunut.



Kuva 4: Lokitietoa graafisesti esitettynä

Jokaisen graafiluokan alussa on kyseisen graafin oleellisten elementtien arvot. Näiden avulla ohjelmaa pystyy tarvittaessa mukauttamaan helposti jokaisen omiin tarpeisiin. Tämä näkyy esimerkiksi kuvista 4 ja 5 niin, että ne käyttävät erilaisia rajoja määrittämään, milloin yhteyksiä on ollut paljon tai vähän. Esimerkiksi jos tahdotaan kehittää kyseisestä ohjelmasta versio pieniruutuisiin laitteisiin kuten matkapuhelimiin, mahdollistavat parametrit graafien pienentämisen. Tämän lisäksi graafit sisältävät tietorakenteet pisteiden sekä palkkien esittämiseen ja palkkien tekstitietojen säilyttämiseen.



Kuva 5: Kuukausi-välilehti

Päivän ja 7. päivän graafit sisältävät aina vakiopituisen ajan sisältä tapahtumia, mutta kuukauden graafin pituus voi vaihdella 28 ja 31 päivän välillä. Kuva 5 esittää kuinka ohjelma piirtää vielä yhden aikavälin 30. päivän jälkeen, sekä sen kuinka yhteenvetopalkkien päälle vietäessä ohjelma näyttää käyttäjälle aikavälin tarkat tiedot. Tämän takia kuukauden graafi sisältää ylimääräistä toimintalogiikkaa, joka pidentää tai lyhentää graafisen esityksen sopimaan valitun kuukauden pituuteen.

7. päivän graafi ei ole rajoittunut oikean kalenterin viikkoihin, vaan mahdollistaa ainoana graafisista välilehdistä vapaavalintaisen aikavälin tarkistelemisen. Päivän graafi on aina keskiyöstä keskiyöhön ja kuukauden graafi ensimmäisen päivän alusta kuun viimeiseen päivään.

4 Testaus

Ohjelman testaus hoidettiin kahdella tapaa. Testauksen runkona toimi ohjelman testaaminen opinnäytetyön kirjoittajan toimesta Windows-alustalla Laurean DSiP-testiympäristöstä saaduilla lokitiedoilla. Tämän lisäksi kirjoittaja testasi ohjelmaa ennen overview-välilehden lisäystä myös Laurean Leppävaaran DSiP-testiympäristössä Linuxilla. Aikataulutussyistä

johtuen ohjelmaa ei testattu enää vuoden vaihteen jälkeen Laurean Linux ympäristössä. Testauksen toisen osion muodosti ohjelman testaaminen Ajecolla.

4.1 Menetelmä

Opinnäytetyön kirjoittajan toteuttamassa testauksessa käytiin läpi ohjelman toimintoja eri lokitiedostoilla ja eri aikarajoilla. Johtuen ympäristön tuoreudesta ja testidatan luonteesta, testattavien tiedostojen sisältämät lokimerkinnät olivat valitettavan lyhyeltä aikaväliltä, joten ohjelman vasteajoista pitkällä aikavälillä ei pystytty vetämään johtopäätöksiä.

Koska ohjelman lopullisista käyttäjistä ei ollut tarkkaa tietoa tai profiilia, jätettiin käytettävyystestaus tekemättä. Ohjelman käytettävyyden arviointi jäi siis Ajecon asiantuntijoiden kommenttien varaan. Tämän lisäksi Linux testausta Laureassa suorittaessa myös Sami Alaverroselta tuli lisäideoita käytettävyyden parantamiseen, kuten ohjelmaan päätyneen yleisnäkymän alkuperäinen idea.

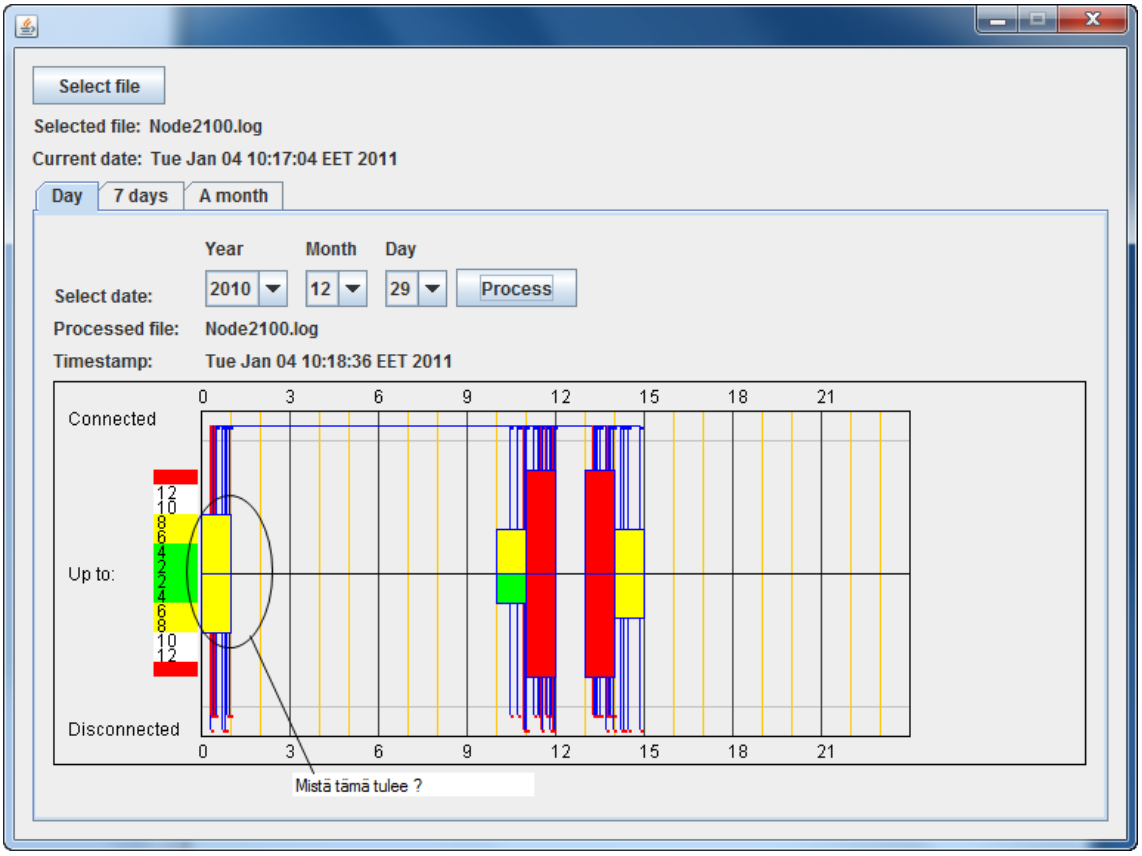
4.2 Havainnot

Lukuun ottamatta muutamaa logiikkavirhettä ohjelma toimi odotetusti sekä Windows- että Linux-ympäristöissä. Linux testeissä ilmeni, että ohjelman täytyy varautua myös siihen että tiedosto voi olla lukittu.

Ohjelmalla oli ongelmia vuoden vaihteessa näyttää kuluvaa vuotta valintamahdollisuuksissa. Tämä johtui siitä, että se laski automaattisesti oletusaikarajakseen viikon taaksepäin nykyisestä hetkestä ja vasta sen jälkeen päätteli mitkä vuodet sen täytyisi laittaa valikkoon.

Ohjelma ei nollannut päivää jos vuosivaihtoehdoksi valitsi ylimpänä listassa olevan vuoden. Päivän nollaaminen on tarpeellista koska näin vältetään tilanne, jossa ohjelma ehdottaisi olematonta päivää valinnaksi.

Kuukausi välilehdellä liikkuminen tab-näppäimellä on rikki.



Kuva 6: Virhetilanne

Testauksen vakavimpana huomiona ohjelmalogiikan pettämisestä tuli Ajecon testeissä ilmi, että ohjelma tallensi kello 12:00-12:59 tapahtuneet tapahtumat 00:00-00:59 kohdalle, kuten kuvasta 6 voi nähdä. Tätä ei ollut tullut ilmi opinnäytetyön kirjoittajan toteuttamassa testauksessa, sillä siinä yleisimmin käytetyssä testitiedostossa ja aikavälissä ei ollut tapahtumia 12:00-12:59 kohdalla, mutta 00:00-00:59 kohdalla oli ollut tapahtumia.

4.3 Korjaukset

Ohjelmaan lisättiin koodia, joka tarkistaa onko lukemisen kohteena oleva tiedosto lukittu. Overview-välilehdellä ohjelma ilmoittaa kohtaamistaan lukituista tiedostoista teksti-ikkunassa tulosten joukossa.

Vuoden vaihteessa ilmenevä ongelma vuoden listaamisessa korjattiin. Päivän nollaamisongelma korjattiin myös.

Kuukausi välilehdellä tab-näppäimellä liikkuminen korjattiin osittain. Liikkuminen on edelleen rikki objektiketjun alusta, mutta loppu ketjun logiikka toimii määritellysti.

Kello 12 tapahtuvat tapahtumat korjattiin oikeaan kohtaan muuttamalla käytettävää päivämäärän muodon määritystä ”hh” muodosta ”HH” muotoon. Näin Java ottaa huomioon myös kello 12 kohdalla sen, että käytetään 24h aikajärjestelmää.

5 Iterointi-vaihe ja lopputulos

Ohjelman toimintalogiikan ja käyttöliittymän ollessa kehitystyön aikana jatkuvassa käytössä, ilmeni mahdollisuuksia kehittää ohjelmaan pieniä käyttöliittymäparannuksia. Hyvänä visuaalisena esimerkkinä näistä on elementtien ryhmittely, kuten vanhempaa versiota edustavan kuvan 6 aikaleimojen eroista muihin kuviin nähden pystyy näkemään. Testausvaiheessa pyysin myös ulkopuolista näkökulmaa Alaverroselta ja Ajecolta.

Ajecolta pyydettiin lisäämään yhteyden tilaa kuvaavien vaakaviivojen paksuutta, jotta ne erottuisivat paremmin taustaruudukosta. Viivojen paksuutta lisättiin toiveiden mukaisesti, kuitenkin niin että varsinaisia tapahtumia kuvaavat pisteet jätettiin paksummiksi.

Alaverrosen kanssa käytyjen keskustelujen perusteella tuli ilmi, että ohjelmassa olisi hyvä olla näkymä, josta saisi koko verkon kattavan kuvan mahdollisista ongelmatapauksista. Koska mielestäni graafinen esitystapa ei soveltunut esitystavaksi ennalta määrittelemättömään määrään nodeja ja palvelimia, ehdotin Ajecon asiantuntijoille, että kyseisen näkymän voisi toteuttaa tekstipohjaisesti. Idea hyväksyttiin ja tämän perusteella ohjelmaan lisättiin Overview-välilehti.

Aluksi Overview-välilehden aikavälin määritys oli jätetty pienemmäksi, koska tulevaisuudessa ohjelma saattaa joutua käymään läpi todella suuren tietomäärän tehdäkseen yhteenvetonsa. Koska ohjelma on kuitenkin tarkoitettu asiantuntijakäyttöön, pidennettiin aikavälin määritystä mahdollistamaan todella pitkän aikavälin yhteenvedot.

Lopullinen versio vastaa hyvin suunnitelmaa, parannettuna Overview-välilehdellä. Kun ohjelmaa kehitettiin jatkuvalla iteroinnilla, myös käyttöliittymästä hioutui entistä käyttäjäystävällisempi. Ohjelma onnistuttiin toteuttamaan Javan vakiokomponenteilla, joten laitteistoriippumattomuus toteutui sitä kautta. Ohjelma ja sen kommentointi toteutettiin englanninkielellä. Tämän lisäksi myös suppea arkkitehtuurikuvaus ja lyhyen käyttöohjeistuksen sisältävä readme-tiedosto toteutettiin englanniksi (Liite 1, Liite 2).

6 Jatkokehitysehdotukset

Tulevaisuudessa voi tulla tarve integroida tämä ohjelma tiiviisti isompaan hallinnointiohjelmaan. Tässä tapauksessa olisi aiheellista harkita tiedostojen lukemiseen

käytetyn toimintalogiikan siirtämistä nappien toimintojen alta omiksi toiminnoikseen. Siinä missä nykyinen rakenne tarjosi vakaan ja turvatun ympäristön ohjelman toiminnalle, tulisi kyseisessä tilanteessa tarve välittää tietoja parametreina.

Toiminnan eriyttäminen mahdollistaisi myös linkityksien luomisen eri välilehtien välille. Näin käyttäjä voisi esimerkiksi klikata hiirellä kuukausivälilehdellä 3. päivän yhteenvetopalkkia ja ohjelma hakisi sen päivän tiedot päivävälilehdelle. Hankkeen loppuarvioinnissa Ajecolta esitettiin samantapainen idea (Holmström 2011).

Toiminnan poistamisessa napin toiminnan suojista pitää kuitenkin ottaa huomioon se, että tämä luo tarpeen lisätarkistuksille toimintalogiikkaan. Tämä johtuu siitä, että nyt ohjelma käytännössä tarjoaa käyttäjälle vaihtoehdot, mitä se hyväksyy syötteinä sisään. Tästä johtuen ohjelma pyrkii varmistamaan syötteiden laadun jo luodessaan niitä, eikä joudu suorittamaan erittäin tarkkaa tarkistusta käyttäessään niitä. Yksin toimiessaan toimintalogiikan täytyisi siis varmistaa entistä tarkemmin, että annetut parametrit ja niiden luomat yhdistelmät ovat hyväksyttyjä.

Ajecolta tulleessa loppuarvioinnissa esitettiin myös, että tehdyt päivämäärä valinnat siirtyisivät automaattisesti toisille välilehdille (Holmström 2011). Vaikka kyseessä on yksinkertaisesti toteutettavissa oleva ominaisuus, se saattaa aiheuttaa myös sekaannusta, kun tehdyt valinnat vaikuttavat käyttäjältä piilossa myös muille välilehdille. Tästä johtuen sen toteuttaminen kannattaa harkita tarkasti laajempaan käyttöliittymätestaukseen perustuen.

Loppuarvioinnissa toivottiin mahdollisuutta asettaa oletushakemisto ympäristömuuttujaan tai asetustiedostoon. Näin käyttäjän ei tarvitsisi valita tiedostoa jostain hakemistosta, ennen kuin hän voi katsoa tuloksia Overview-välilehdeltä. (Holmström 2011.) Asia oli tullut esille jo kehitystyötä tehdessä, sillä se olisi myös nopeuttanut ohjelman käynnistyksen jälkeistä toimintaa myös muilla välilehdillä. Se päätettiin jättää kuitenkin pois toistaiseksi, sillä se olisi todennäköisesti rikkonut ohjelman toteuttaman täydellisen laitteistoriippumattomuuden. Koska laitteistoriippumattomuus oli otettu yhdeksi tärkeimmistä toteutuskriteereistä, ei siitä haluttu luopua.

Loppuarvioinnissa pohdittiin ovatko graafin pystyviivat hieman rauhattomia, jos tapahtumia on paljon. Samalla tiedostettiin, että testimateriaalissa oli tarkoituksellisesti huomattavasti enemmän tapahtumia kuin todelliseen toimintaan pohjautuvassa materiaalissa. (Holmström 2011.) Tämä on käytännössä tarkoituksellista, sillä ohjelman tarkoitus on nostaa esille mahdollisesti ongelmalliset kohdat.

7 Arviointi

Ohjelma toteuttaa hyvin sille asetetut tavoitteet: lokitiedostojen visualisoiminen päivä-, viikko- ja kuukausitasolla; vakaa ja virheetön toiminta.

”Ohjelmaa on koekäytetty ja se piirtää DSiP-järjestelmän lokitiedostojen pohjalta ymmärrettävän graafin, joka auttaa käyttäjää näkemään noden kytkeytymistiheyden.” Ajeco Oy:ltä saadun arvion mukaan ohjelmointityö on toteutettu hyvin ja graafinen käyttöliittymä on hyvätasoinen. ”Katsomme osaltamme, että tehty työ on ONT:n kannalta kiitettävä.” (Holmström 2011).

Oman ammatillisen kehittymisen kannalta kaikki tavoitteet toteutuivat. Opin Javan graafisen käyttöliittymän toteutuksesta sekä 2D-elementtien kanssa työskentelystä, onnistuin kommentoimaan koodia järjestelmällisesti sekä selkeästi ja toteutin ensimmäistä kertaa kattavan virheentarkistuksen Javan tavalla hoitaa asia.

”Mikko on ansiokkaasti itse suunnitellut graafisen työn layoutin ja koodannut ohjelman Javalla. Arviomme on, että Mikko on suorittanut ohjelmointityön hyvin ja on kehityskelpoinen ohjelmistokehitysalaa ajatellen.” Työ on myös toteutettu itsenäisesti ja Ajeco:lta annettuja kommentteja huomioon ottaen. (Holmström 2011.)

8 Yhteenveto ja johtopäätökset

Tutkimustieteellisen lähestymistavan avulla opinnäytetyö ja siinä valmistunut ohjelma onnistuivat erittäin hyvin. Ohjelma toimii halutulla tavalla ja ratkaisee ajankohtaisen yritysmaailman ongelman. Sen käyttökelpoisuus ja laatu on todennettu mahdollisuuksien sallimilla testauksilla. Se on kehitetty täsmällisesti Javan kehittäjän ohjeistuksen ja esimerkkien mukaisesti. Lähestymistavan mukaisesti tuotos syntyi iteratiivisen prosessin tuloksena, jossa kehittäjän oppimisen ja toimeksiantajan ohjeiden mukaisesti ohjelmaa muokattiin entistä sopivammaksi tarkoitukseen. Silti ohjelmaa kehittäessä pysyttiin ongelma alueen rajojen sisäpuolella, eikä lähdetty toteuttamaan mahdolliseen jatkokehitykseen paremmin soveltuvia osa-alueita. Työ on myös pyritty esittämään tehokkaasti niin liiketoiminnan hallintaan suuntautuneelle yleisölle tässä opinnäytetyöraportissa, kuin tekniikkaan suuntautuneille ihmisille etenkin Java-koodin kommentoinneissa.

Tämä opinnäytetyö osoitti, että Java soveltuu erittäin hyvin ammattimaiseen laitteistoriippumattomaan koodaukseen, jossa lopputuotos toteuttaa vakioelementtejä laajentavan graafisen käyttöliittymän. Tämän lisäksi opinnäytetyötä tehdessä tuli esille se kuinka opinnäytetyössä toteutettavan ohjelman toiminta ja rajaus on hyvä valita niin, että se

ei ole keskeisenä osana kehitettävän järjestelmän ydintoimintaa. Näin ohjelmaa pystyy työstämään itsenäisesti ja omaan tahtiin ilman, että olisi täysin riippuvainen kokonaisuuden edistymisestä.

Lähteet

Hevner, A., March, S., Park, J. & Ram S. 2004. Design Science in Information Systems Research. MIS Quaterly Vol. 28, 75-105.

Holmström J. 2010. DSiP-luento Laureassa 30.9.2010. Ajeco Oy.

Holmström J. 2011. Loppukommentit 16.2.2011. Ajeco Oy.

Holmström J. & Ramstedt, K. 2010. Sähköpostikeskustelu 16.11.2010. Ajeco Oy.

Oracle. Java Technology. Viitattu 11.12.2010
http://www.oracle.com/us/technologies/java/index.html

Oracle. What is Java Technology and why do I need it? Viitattu 11.12.2010.
http://www.java.com/en/download/faq/whatis_java.xml

Kuvat ja kuviot

Kuva 1: Ohjelman rakenne 9
Kuva 2: Aloitusnäkymä 10
Kuva 3: Overview-välilehti 11
Kuva 4: Lokitietoa graafisesti esitettynä 12
Kuva 5: Kuukausi-välilehti 13
Kuva 6: Virhetilanne 15

Liitteet

LIITE 1: DSiPConnectionCounter Architecture

LIITE 2: DSiPConnectionCounter Readme

Classes:
DSiPConnectionCounter
Day_DrawPanel
Week_DrawPanel
Month_DrawPanel
DSiPLogFileFilter

The body of the program is the DSiPConnectionCounter class. It handles the logic for selecting the file/folder and for extracting the information from it. It is also responsible for the GUI, except for the Graphs. The tooltips are created with the help of the graph classes. Since the overview tab doesn't actually draw anything, all of the logic for it is under the DSiPConnectionCounter.

The 3 graphs for Day, Week and Month are handled by their respective classes. They contain the information on how to draw the graphs as well as do the actual drawing. Before they can draw the graph, they need the DSiPConnectionCounter to feed them the extracted information from the files.

The DSiPLogFileFilter is used by the DSiPConnectioNCounter to filter files in the file chooser box. This way users can only process valid DSiP node and router files.

DSiP Connection Counter - readme

***
IMPORTANT NOTE: DSiP Connection Counter properly reads only logs
that follow the timestamp format of "dd.mm.yyyy".
***

DSiP Connection Counter is a tool to analyze your DSiP logs from your
DSiP-servers and DSiP-nodes. It allows you to choose the time period
in three different kinds of ways.

The dots are an accurate representation of the hour the event happened
during but they are only an approximation of the minutes. The lines
represent whether the node was online or offline, but they do not
extend past the timeframe. So for example a node that was online
before the timeframe started, and is still online, is shown as
starting with an online dot, instead of a line coming from the past.
This applies to the last dot as well.

Day view

Displays the events from the log for a one day period. For each hour
a bar for total connections and disconnections is shown if that
timeframe had any events.

7 day view

This view allows you to see the events of the log from a 7 day period.
This is not restricted to actual weeks, but you can choose any 7 day
period that stars within the last 10 years. This also makes it the only
view that allows you to see past this year.

For situations where you are viewing log files from a different
timezone, this is the only tool that lets you see the records from
"the future" when the year is changing. This is the only view that
skips the bigger timeframe when displaying the total bars. Instead

of showing a total bar for every day, it displays a total bar for every 12 hours.

Month view

This view allows you to see the events of the log from a period of a month. This view always starts on the 1st day and ends on the last day. The total bars show the connections and disconnections of the whole day.

Overview

This tab allows you to see in which log files were there more events (connections/disconnections/being disconnected) than you set as the limits for that timeframe.