TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Markus Latvala

TAPAHTUMANKERÄYSJÄRJESTELMÄ

Työn ohjaaja Ohjelmistotekniikan lehtori Tony Torp
Työn teettäjä Netland Oy, valvojana Pekka Ahmavuo
Tampere 2005

Tekijä:	Markus Latvala
Työn nimi:	Tapahtumankeräysjärjestelmä
Päivämäärä:	
Sivumäärä:	37 sivua ja x liitesivua
Hakusanat:	
Koulutusohjelma:	Tietotekniikka
Suuntautumisvaihtoehto:	Ohjelmistotekniikka
Työn valvoja:	Ohjelmistotekniikan lehtori Tony Torp
Työn ohjaaja:	Tekninen asiantuntija Pekka Ahmavuo Netland Oy, Tampere

Yrityksillä on usein käytössä monipuolisia järjestelmiä jotka koostuvat eri valmistajien toimittamista komponenteista ja ohjelmistoista. Eri laite- ja ohjelmistovalmistajien välillä ei kuitenkaan ole sovittu minkäänlaisia tapahtumia käsitteleviä standardeja ja jokaisella valmistajalla onkin oma toteutuksensa. Erilaisia tapahtumia ovat virhetilanteet tai vaikkapa yhteystapahtumat. Olisi tärkeää saada kaikkien käytössä olevien laitteiden ja ohjelmien tuottamat tapahtumat keskitettyä yhteen tietokantaan josta niitä voitaisiin valvoa ja käsitellä helposti.

Insinöörityön tarkoituksena oli suunnitella ja toteuttaa järjestelmä jonka avulla yrityksen kaikkien käytössä olevien komponenttien ja ohjelmistojen virheilmoitukset voitaisiin tallentaa. Järjestelmään tallentuvia tapahtumia voidaan ohjata monipuolisten sääntöjen mukaan ja myös lähettää tapahtumia käyttäjän kännykkään tai sähköpostiin.

Järjestelmien valvominen ilman kunnon työkaluja on erittäin työlästä ja usein tärkeät ilmoitukset jäävät huomaamatta ja korjaustoimenpiteet saattavat viivästyä. Lisäksi uusien työntekijöiden kouluttaminen järjestelmiä valvomaan on hankalaa, koska monien eri komponenttien tapahtumatietojen ymmärtämiseen menee helposti pitkä aika.

Ohjelmointikieleksi toteutukseen on valittu Perl ja käyttöliittymän rakennustyökaluksi Netland Oy:n kehittämä AppRunner-sovelluspalvelinratkaisu. Molemmat toimivat GNU/Linux-käyttöjärjestelmässä ja tämä yhdistelmä on todettu toimivaksi Netland Oy:n tarpeisiin ja otettiin käyttöön myös tähän projektiin.

tapahtumankeräysjärjestelmällä pyritään helpottamaan ja nopeuttamaan työntekijöiden valvontatoimia sekä lyhentämään virhetilanteisiin liittyviä reagointiaikoja ennen korjaustoimenpiteiden aloittamista. Järjestelmän ansiosta reagointiajat ovat lyhentyneet ja valvojat saavat myös tärkeimmät hälytykset halutessaan tekstiviestinä tai sähköpostina. Järjestelmän käyttäjillä on myös mahdollisuus esimerkiksi kännykän avulla kuitata järjestelmän lähettämä tapahtuma käsitellyksi tai työn alle otetuksi.

Tulevaisuudessa järjestelmää haluttaisiin vielä hieman yleiskäyttöisemmäksi, että siihen liitettävien uusien laitteiden ja sovellusten tapahtumatietojen tuominen järjestelmään sujuisi mahdollisimman mutkattomasti. Lisäksi sääntöjen käsittelyn käyttöliittymäkomponenttia pitäisi kehittää käyttäjäystävällisemmäksi ja intuitiivisemmaksi. Tällä hetkellä käyttöliittymä on hieman liian hajanainen ja epäselvä.

Author of thesis:	Markus Latvala
Name of thesis:	Event Collector, Monitor and Forwarder
Date:	
Engineering thesis:	37 pages, x appendices
Keywords:	
Degree Programme:	Computer Systems Engineering
Specialisation:	Software Engineering
Thesis Supervisor:	Lecturer Tony Torp
Commissioning Company:	Technical Expert Pekka Ahmavuo Netland Oy, Tampere

Companies often have many different information systems in use at the same time. These systems are manufactured by different hardware and software companies. All systems produce some sort of events, be they error messages or merely different kinds of event messages. Mostly these messages have nothing in common with each other and thus are difficult to monitor efficiently. It would be ideal, if these message could be gathered into one central database in a uniform way so that monitoring these message would be easier and more efficient. The aim for this thesis was to produce a way of collecting these messages into a central database and provide a way of monitoring and controlling the messages. Perl programming language was used to create the software components and AppRunner software server for the user interface.

ALKUSANAT

Olen opiskellut Tampereen ammattikorkeakoulussa tietotekniikan koulutusohjelmassa vuodesta 1999 alkaen.

Valitsin suuntautumisvaihtoehdokseni ohjelmistotekniikan.

Insinöörityöni olen tehnyt ohjelmistosuunnittelijana, Netland Oy:ssä Vammalassa. Työn aikaansaannoksena olen suunnitellut ja toteuttanut järjestelmän, jonka on tarkoitus tulla käyttöön Etelä-Satakunnan puhelin Oy:lle ja myöhemmin mahdollisesti myös muille Netland Oy:n asiakkaille.

Tämän lopputyöni ohjaajana toimi ohjelmistotekniikan lehtori, Tony Torp.

Tampereella huhtikuussa 2005

Markus Latvala

SISÄLLYSLUETTELO

KÄYTETYT TERMIT JA LYHENTEET

Apprunner	Netland Oy:n kehittämä sovelluspalvelinratkaisu. Käytetään monipuolisten verkko- ja mobiilipalveluiden toteuttamiseen. ks. Kappale 3.1.1
HTTP	Hypertext Transfer Protocol. Ohjelmistotason tiedonsiirto-protokolla jonka välityksellä internet-selain ja WWW-palvelin keskustelevat.
HTML	Hypertext Markup Language. World Wide Web Consortium:n kehittämä julkaisukieli internet-sivustoille. Perustuu aikaisempaan painotason julkaisukieleen SGML:n.
DX 200	Digital Telephone Exchange. Nokian valmistama, Euroopassa aikoinaan ensimmäinen täysin digitaalinen puhelinkeskus.
Tapahtumakerääjä	Tapahtumakerääjän osasovellus. Kerää tapahtumia eri lähde-laitteista ja -sovelluksista, muuntaa tapahtumat tapahtuma-monitorin ymmärtämään muotoon sekä lähettää tapahtumat monitorisovelluksen tietokantaan HTTP-rajapinnan kautta.
Tapahtumamonitori	Tapahtumakerääjän osasovellus. Tarjoaa käyttöliittymän tapahtumien selaamiseen ja käsittelyyn, tallennussääntöjen määrittämiseen sekä lähetyslistojen muodostamiseen. Sovelluksesta voi myös tarkastella lähetystä odottavia edelleenlähetyksiä.
Sääntökone	Tapahtumamonitorin osa, käy läpi käyttäjän määrittelemän sääntölistan sekä suorittaa toimenpiteitä tapahtumalle säännöissä määriteltyjen ehtojen ja toimintojen mukaan.
Tapahtumalähettäjä	Tapahtumakerääjän osasovellus. Lähettää tapahtuman eteenpäin esim. tekstiviestillä tai sähköpostilla määriteltyyn osoitteeseen/numeroon.

Tapahtumakanta	tapahtumankeräysjärjestelmän osasovelluksien, monitorin ja tapahtumalähettäjän tietokantataulut sisältyvät tapahtumakannan määrittelyyn.
ADSL	Asynchronous Digital Subscriber Line. Tiedonsiirtotekno-logia joka mahdollistaa nopeiden tietoliikenneyhteyksien muodostamisen vanhoja kuparisia puhelinjohtoja käyttäen.
Perl	Practical Extraction and Report Language. Tulkattava ohjelmointikieli, jota käytetään pääasiassa ns. Liimakielenä erilaisten järjestelmien ja rajapintojen yhdistämisessä. Myös monet nykypäivän www-sivut on toteutettu Perl-kielen avulla.

1 JOHDANTO

Insinöörityön tarkoituksena oli suunnitella ja toteuttaa järjestelmä, jonka avulla Etelä-Satakunnan Puhelin Oy:n työntekijät voivat valvoa eri järjestelmien tuottamia tapahtuma- ja hälytystietoja helposti yhdestä paikasta yhtenäisessä muodossa.

Aikaisemmin eri järjestelmien valvonta on hoidettu nk. käsipelillä ja yleensä käytetyt valvontatekniikat ovat kömpelöitä ja hankalia oppia. Esimerkiksi Nokialta hankitun DX 200 digitaalisen kytkinjärjestelmän tapahtumatietoja on valvottu yksinkertaisella telnet-yhteydellä, jossa tiedot virtaavat jatkuvasti näytöllä hankalasti ymmärrettävässä tekstimuodossa.

Toteutetulla järjestelmällä haluttiin helpottaa Etelä-Satakunnan Puhelin Oy:n työntekijöiden valvontatoimia ja saada nopeutettua havaittuihin tapahtumiin reagointia. Sovelluksen ansiosta myös uusien valvojien kouluttaminen helpottuu huomattavasti, koska kaikkien käytössä olevien järjestelmien tapahtumat saadaan tallennettua yhtenäiseen helposti ymmärrettävään muotoon.

Järjestelmästä pyrittiin tekemään mahdollisimman yleiskäyttöinen, jolloin erilaisten tapahtumalähteiden liittäminen järjestelmään olisi mahdollisimman yksinkertaista. Kuitenkin järjestelmä suunniteltiin alun perin toimimaan mahdollisimman hyvin Nokian DX 200 kytkinjärjestelmän ja siitä muodostuvien tapahtumien kanssa, koska tämän järjestelmän tapahtumien seuranta on toimeksiantajayritykselle ensisijaisen tärkeää.

Valvontasovelluksen käyttöliittymän haluttiin olevan mahdollisimman helposti käytettävissä mistä tahansa ja siksi kehitysympäristöksi valittiin web-ympäristö. Web-ympäristön ansiosta valvontasovellus on käytettävissä kaikkialla missä on käytössä HTML 4.01 standardia tukeva www-selain ja internet-yhteys. Tällainen alustariippumattomuus takaa erinomaisen toimintavalmiuden eikä vaadi käyttäjältä asennustoimenpiteitä. Myöhemmin järjestelmästä olisi pienillä lisäyksillä ja muutoksilla myös toteutettavissa esimerkiksi mobiililiittymä WAP-käyttöä ajatellen.

2 TOIMEKSIANTAJA /2/

Kappaleessa esitellään lyhyesti työn toimeksiantajan historiaa ja liiketoimintamallia.

Etelä-Satakunnan Puhelin Oy (ESP) on toiminut Vammalan kunnassa vuodesta 1893 ja sen toimialueeseen kuuluvat tämän lisäksi Mouhijärven ja Suodenniemen kunnat sekä Kiikka Äetsän kunnasta. Alun perin yrityksen nimi oli Tyrvään Telefoni Osakeyhtiö kunnes 50-luvun alussa yrityksen toimilupa siirrettiin Vammalan Seudun Puhelin Oy:lle. Mouhijärven Puhelin Oy ja Vammalan Seudun Puhelin fuusioituivat vuonna 1959 ja vuonna 1974 yrityksen nimi vaihdettiin Etelä-Satakunnan Puhelin Oy:ksi. Nykyisin yrityksen toimialueella asuu jo n. 23 000 henkilöä. Etelä-Satakunnan Puhelin käynnisti kaapelitelevisioverkon rakennustyöt vuonna 1981 ja nykyisin yritykselle kuuluu noin 3000 kaapeli TV-taloutta.

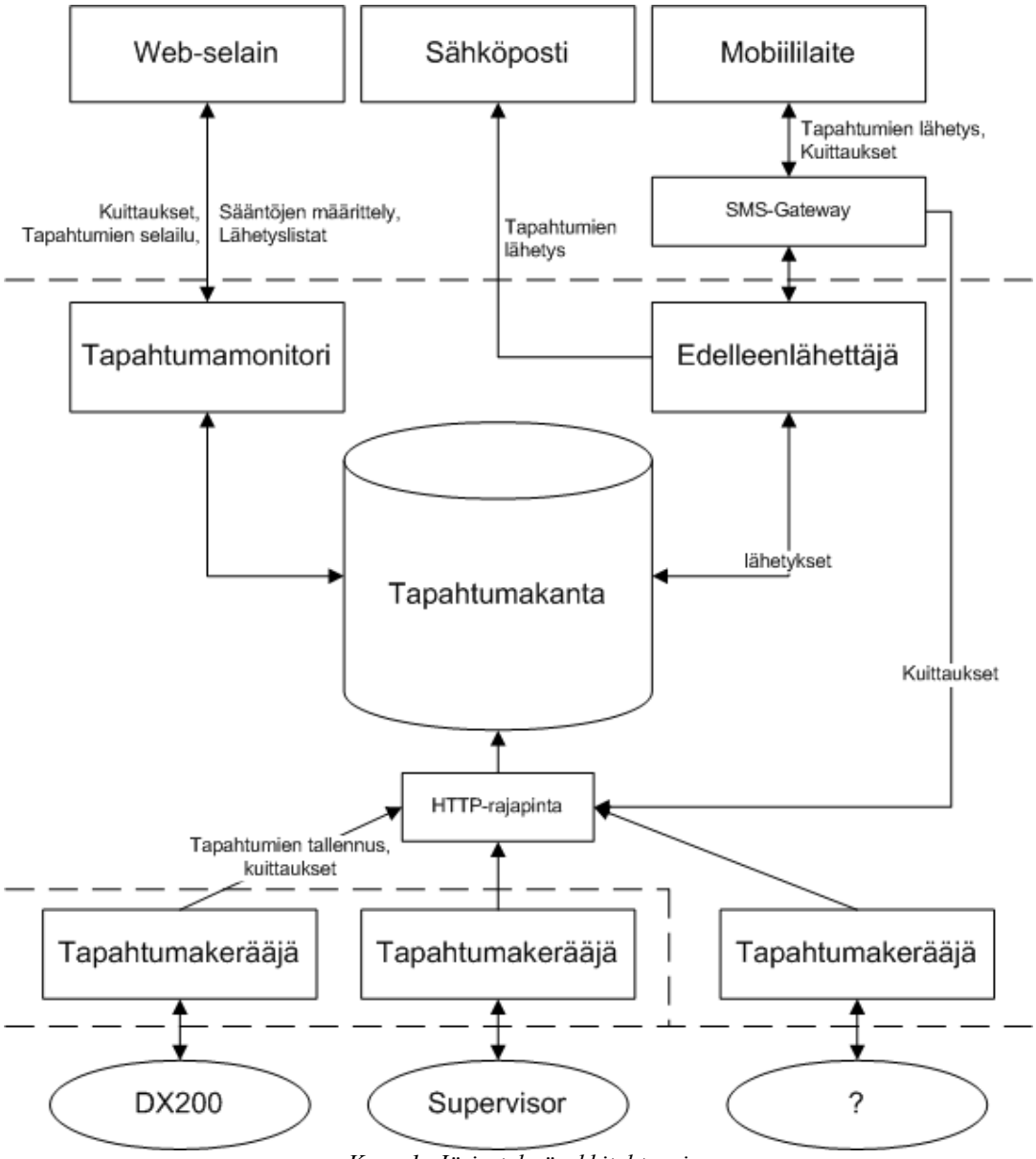
ESP on toiminut tavallisena asiakkaidensa omistamana puhelinyhtiönä ja vuonna 1999 Etelä-Satakunnan Puhelin aloitti myös Kopteri-internet palvelun ja toimii edelleen alueen ainoana laajakaistaliittymien tarjoajana. Vuoden 2003 lopussa kopterin liittymiä oli toimialueella 2071 kappaletta joista 867 kappaletta oli ADSL-liittymiä.

Yrityksen toimipiste sijaitsee Vammalan kaupungin keskustassa, puhelinyhtiön talosta. Asiakkaina ESP:llä ovat yksityishenkilöt sekä pienet- ja keskisuuret yritykset. Henkilökuntaa oli palkkalistoilla 49 kappaletta vuonna 2003. Samana vuonna liikevaihto oli 6,94 m€ kun investoinnit olivat 1,08 m€.

Netland Oy ja Etelä-Satakunnan Puhelin Oy ovat olleet tiiviissä yhteistyössä jo vuosien ajan, ja siksi tämä projekti luotettiin Netland Oy:n toteutettavaksi.

3 JÄRJESTELMÄN TOIMINTA

Kappaleessa kuvataan järjestelmää yleisellä tasolla ja esitellään järjestelmän eri osa-alueet sekä puhutaan hieman kehitys- ja toimintaympäristöstä. Kuva 1 esittelee järjestelmän rakenteen ja siitä nähdään perusajatus josta lähdettiin liikkeelle kokonaisuuden suunnittelussa.



Kuva 1: Järjestelmäarkkitehtuuri

Järjestelmä koostuu kolmesta osasovelluksesta joilla on kaikilla tehtävänsä kokonaisuuden toiminnassa. Osa-alueet ovat kerääjä-, monitori- ja

tapahtumalähettäjäsovellukset. Tapahtumamonitori ja tapahtumalähettäjä toimivat samassa koneessa, kun taas tapahtumakerääjät voivat olla toiminnassa fyysisesti eri laitteissa kuin monitori ja tapahtumalähettäjä. Kerääjiä voi siis olla ajossa useita kappaleita samaan aikaan eri sijainneissa ja kaikki tallettavat keräämänsä tapahtumat samaan tapahtumamonitorin tapahtumakantaan yhteisellä HTTP-rajapinnalla. Tarkemmat selitykset osasovelluksien toiminnasta löytyy kappaleista 3.3, 3.4 ja 3.5.

3.1 Kehitys- ja toimintaympäristö

Sovellusten toiminnallisuus toteutettiin perl-ohjelmointikielellä, sekä Netland Oy:n kehittämällä Apprunner-sovelluspalvelinratkaisulla. Apprunner-sovelluspalvelin tarvitsee toimiakseen Apache-projektin kehittämän www-palvelimen. Tuttujen työkalujen ja sovellusten käyttäminen järjestelmän toteutuksessa vaikuttivat suuresti kehitysympäristön valintaan. Perl on oivallinen valinta senkin takia, että suurin osa kaikesta tapahtumatiedoista saadaan tekstimuodossa ja Perl sisältää erittäin tehokkaat työkalut tekstin käsittelemiseen ja muokkaamiseen. Perl on myös täysin olio-ohjelmointiin valmis kieli, vaikkei se tarjoakaan samanlaisia luokkarakenteita kuten esim. C++ tai Java. Perlin moduulirakenne kuitenkin luo puitteet periyttämiselle ja kapseloinnille, joten monipuoliset dynaamiset luokkarakenteet ovat toteutettavissa myös perlin avulla.

Apprunner-sovelluspalvelin oli luonnollinen valinta monitorisovelluksen käyttöliittymän toteutukseen, koska käyttöliittymä päätettiin tehdä HTML-toteutuksena ja Apprunner on parhaimmillaan juuri web- ja mobiilisovelluksien toteutuksissa. Apprunneria käytettiin myös toteuttamaan HTTP-rajapinta tapahtumien tallennusta varten.

Käyttöliittymän toteuttaminen web-ratkaisuna on erittäin käytännöllinen myös siitä syystä, että liittymä on käytettävissä kaikissa tietokoneissa ja käyttöjärjestelmissä joissa on käytössä HTML 4.01 standardia tukeva www-selain. Tapahtumamonitorin käyttöliittymä ei vaadi käyttäjältä asennustoimenpiteitä jos järjestelmän www-

selain on toimintakunnossa.

Palvelinkoneen käyttöjärjestelmänä toimii Debian GNU/Linux r3.1 Sarge -Linux levitysversio. Projektissa käytetyn Apache www-palvelimen versionumero on 1.3.31 ja Perl kielen versio on 5.8. Tietokantasovelluksena käytössä on Postgresql, versio 7.2.

3.1.1 AppRunner-sovelluspalvelin /3/

AppRunner-sovelluspalvelin on Netland Oy:n kehittämä tehokas alusta verkko- ja mobiilipalveluiden toteuttamiseen. Sovelluspalvelin perustuu olioarkkitehtuuriin, minkä ansiosta AppRunnerilla toteutetut sovellukset ovat erittäin muuntautumiskykyisiä ja helposti laajennettavissa. AppRunnerin yksi pääperiaatteista onkin, että sovellusten suorittaminen jätetään tehokkaiden keskuspalvelimien (http-palvelin) huoleksi jolloin sovelluksista saadaan selain- ja alustariippumattomia. Tämä periaate mahdollistaa käyttöliittymien toteuttamisen hyvin erityyppisiin ympäristöihin, kuten www-selaimeen tai wap-puhelimeen.

AppRunner myös edistää verkko- ja mobiilihankkeiden nopeata kehitystä erottamalla ohjelmoijien ja käyttöliittymäsuunnittelijoiden työroolit toisistaan. Ohjelmoijat rakentavat sovellusten toiminnalliset komponentit ja suunnittelijat yhdistävät ne erilaisiin käyttöliittymiin yksinkertaisella AppRunner-kuvauskielellä.

3.2 Tapahtumalähteet

Etelä-Satakunnan Puhelin Oy tarjoaa monenlaisia tietoliikennepalveluja asiakkailleen aina lankapuhelimesta internet-yhteyksiin ja siksi yrityksen käytössä olevat järjestelmät ovat erittäin monimuotoiset ja kirjavat. Yksi suurimmista ongelmista tällaisten monimuotoisten järjestelmärakenteiden kanssa on laitteiden erilaisten tapahtumatietojen kerääminen ja valvominen luotettavasti sekä tehokkaasti. Lisäksi suurin osa käytettävistä laitteista luovuttaa tapahtumatietonsa erittäin hankalasti ymmärrettävässä tekstimuodossa joka vaikeuttaa valvontaa

ratkaisevasti. Toimeksiantajalla on kaksi järjestelmää johon keräysjärjestelmä ensisijaisesti suunniteltiin. Tarkastellaan näitä kahta järjestelmää hieman tarkemmin.

Koska tapahtumalähteitä on useita, määritellään jokaiselle tapahtumalähteelle oma luokkansa, jolloin monitorisovelluksessa voidaan helposti hakea eri luokkien tapahtumia, eikä tapahtumien selailussa eri lähteiden tapahtumat sekoitu keskenään. Määrittämällä luokat eri tapahtumalähteille voidaan muodostaa sääntöketjut jokaiselle luokalle erikseen.

3.2.1 DX 200

DX 200 on digitaalinen puhelinkeskus jota käytetään valvomaan monenlaisia liittymiä aina puhelinliittymistä ADSL-yhteyksiin. DX 200:n virheilmoituksien seuranta ja niihin nopea reagoiminen ovat elintärkeitä asiakkaiden liittymien ylläpidon kannalta. DX 200 tarjoaa tapahtumatietonsa telnet-yhteyden välityksellä monirivisinä viesteinä joille on määritelty suhteellisen tarkka formaatti. Listauksessa 1 on muutama esimerkkitapahtuma muodossa jossa DX 200 syöttää tapahtumia telnet-yhteyden välityksellä.

```
           DX220-EPI2       EVAR  OMU         SWITCH    2003-11-28  13:53:58.21
*** ALARM  SUB-5-75   1A063-37    RCXPRO
    (8948) 2699 CRITICAL UNIT IN INCORRECT WORKING STATE
                BL-EX


           DX220-EPI2       EVAR  OMU         SWITCH    2003-11-28  13:54:00.17
    NOTICE SUB-5-75   1A063-37    RCXPRO
           0691 AUTOMATIC RECOVERY ACTION
                BL-EX BL-ID C000 0003 0003 0000 0000 0000


           DX220-EPI2       EVAR  OMU         SWITCH    2003-11-28  13:54:04.17
    NOTICE SUB-5-75   1A063-37    RCXPRO
           0690 WORKING STATE CHANGE
                BL-ID WO-EX 4C02 0345 0881 0000 0000 0000


           DX220-EPI2       EVAR  OMU         SWITCH    2003-11-28  13:54:04.28
... CANCEL SUB-5-75   1A063-37    RCXPRO
    (8948) 2699 CRITICAL UNIT IN INCORRECT WORKING STATE
                BL-EX


           DX220-EPI2            LSU-1       SWITCH    2003-11-28  14:00:53.46
    NOTICE LSU-1      1E001-37    SI9PRB
           0113 OUTGOING CIRCUIT BLOCKED
                439d 1d 08 0000 0d 0d 020F 017B


           DX220-EPI2            LSU-1       SWITCH    2003-11-28  14:03:03.78
    NOTICE LSU-1      1E001-37    SI9PRB
```

```
           0113 OUTGOING CIRCUIT BLOCKED
                271d 19d 08 0000 0d 0d 020F 017B


           DX220-EPI2             LSU-1       SWITCH    2003-11-28  14:03:24.68
**  ALARM  LSU-1      1E001-37    SI9PRB
    (8949) 2163 ABNORMAL LINE SIGNAL RECEIVED
                433d 38d 24d A0 09


           DX220-EPI2             LSU-1       SWITCH    2003-11-28  14:04:38.78
..  CANCEL LSU-1      1E001-37    SI9PRB
    (8949) 2163 ABNORMAL LINE SIGNAL RECEIVED
                433d 38d 24d A0 09
```

Listaus 1. DX 200-purkin tapahtumalistaus

Yksi tapahtuma vie listauksessa neljä riviä. Ensimmäisellä rivillä on tiedot tapahtuman muodostaneesta laitteesta (DX220-EPI2), tapahtuman sijainnista (EVAR), tyypistä (SWITCH) sekä aikaleimasta. Toisella rivillä ensimmäisenä on mahdollinen tasotieto (**/..), tapahtuman tyyppi (ALARM) ja lähdelaitteen tietoja, jota keräysjärjestelmän kannalta ovat turhia. Kolmannella rivillä on mahdollinen sekvenssinumero (8948), tapahtumakoodi (2699) ja tapahtuman otsikko (CRITICAL UNIT IN INCORRECT WORKING STATE). Viimeisellä rivillä on tapahtuman hexamuotoinen tarkennetieto.

3.2.2 Supervisor

Toinen tapahtumalähde on Supervisor-sovellus joka tallentaa tilatietoja useilta erityyppisiltä antureilta (lämpömittarit, pinnankorkeusmittarit, jne.) ja muodostaa tapahtumia käyttäjän määrittelemien raja-arvojen ylityshetkinä. Esimerkiksi jos lämpötila nousee yli 30 asteen, muodostaa sovellus yksirivisen tapahtuman ylityksestä yksinkertaiseen tekstilogiin. Kun lämpötila laskee takaisin alle 30 asteen raja-arvon, kuittaa sovellus tapahtuman palautumisen normaalitilaan. Kerääjä valvoo sovelluksen tapauksessa tekstilokia, muuntaa sinne ilmestyneet tiedot monitorille sopivaksi ja tallentaa tapahtumarivit kantaan HTTP-rajapinnalla. Listauksessa 2 on muutama esimerkkirivi supervisor-sovelluksen lokista.

```
HÄLYTYS Pri 1 04.08.03. 18:00   Lämpötila              LIK 3           1
NORMAL  Pri 1 04.08.03. 18:02   Lämpötila              LIK 3           0
HÄLYTYS Pri 1 04.08.03. 18:11   Lämpötila              LIK 3           1
NORMAL  Pri 1 04.08.03. 20:51   Lämpötila              LIK 3           0
HÄLYTYS Pri 1 04.08.03. 20:55   Lämpötila              LIK 3           1
NORMAL  Pri 1 04.08.03. 21:11   Lämpötila              LIK 3           0
ACK     05.08.03. 14:21 ESP huolto
        HÄLYTYS Pri 1 15.07.03. 14:14   Lämpötila              LIK 3
ACK     05.08.03. 14:21 ESP huolto
        NORMAL  Pri 1 15.07.03. 15:28   Lämpötila              LIK 3
ACK     05.08.03. 14:21 ESP huolto
        HÄLYTYS Pri 1 15.07.03. 16:03   Lämpötila              LIK 3
```

Listaus 2. Supervisor-sovelluksen tapahtumaloki

Ensimmäisellä listauksen rivillä näkyy Supervisor-sovelluksen tuottama tapahtuma. Tapahtumasta kerrotaan sen prioriteetti ”Pri 1”, aikaleima, tapahtuman kuvaus (joka voidaan Supervisor-sovelluksessa määritellä itse), sijaintitieto sekä tilakoodi. Toisella rivillä on tapahtumaan tullut kuittausilmoitus lämpötilan palautumisesta normaaliksi. Kaksiriviset ACK-tapahtumat ovat tämän tapahtumakeräys-järjestelmän kannalta turhaa tietoa, koska ne tarkoittavat sitä että Supervisorin käyttäjä on kuitannut nähneensä sovelluksen muodostamat tapahtumat.

3.3 Tapahtumakerääjä

Tapahtumakerääjiä voi koko järjestelmässä olla käynnissä useita ja jokaisella kerääjällä on oma hälytyslähteensä. Tapahtumalähteille toteutetaan omat kerääjäluokat jotka muuntavat tapahtumalähteeltä saadun tapahtumatiedon monitorin ymmärtämään muotoon.

Kerääjä lukee lähdelaitteen tapahtumia ja muuntaa saamansa tapahtumatiedot tapahtumankeräysjärjestelmän – tarkemmin sanottuna monitorisovelluksen tapahtumakannan – ymmärtämään muotoon. Muuntamisen jälkeen kerääjä lähettää saadun informaation HTTP-rajapinnan kautta monitorisovellukselle tietokantaan tallennettavaksi. Kaikki eri lähdelaitteiden kerääjät käyttävät tätä samaa rajapintaa tapahtumien tallennukseen.

Jos tietokantakoneeseen (monitorisovellus) ei saada välittömästi yhteyttä, jäävät tapahtumat jonoon levylle spool-hakemistoon, josta ne lähetetään eteenpäin heti kun verkkoyhteydet jälleen toimivat ja monitorisovellus ilmoittaa tapahtuman

tallennuksen onnistuneeksi. Jokaiselle tapahtumalle muodostetaan oma XML-tiedosto spool-hakemistoon.

Kerääjä pitää yllä kahta eri lokitiedostoa. Virhelokiin kerääjä tallentaa kaikki eteen tulleet tunnetut virhetilanteet tapahtumien lukemisesta tallentamiseen, kun taas tapahtumalokiin kirjoitetaan tapahtumien tiedot XML-muodossa kaikista tapahtumakantaan onnistuneesti lähetetyistä tapahtumista.

3.4 Tapahtumamonitori

Tapahtumamonitori on järjestelmän varsinainen käyttöliittymä. Liittymän kautta käyttäjä voi ylläpitää sääntökoneen sääntöjä jotka vaikuttavat tapahtumien tallennukseen ja mahdolliseen tapahtuman jatkolähetykseen. Käyttöliittymän kautta käyttäjä voi myös ylläpitää lähetyslistoja, joihin mahdolliset jatkolähetykset ohjataan. Käyttöliittymä tarjoaa myös mahdollisuuden tarkastella tapahtumia jotka ovat jonossa odottamassa lähetystä. Tärkein käyttökohde monitorille sääntöjen käsittelyn rinnalla löytyy mahdollisuudesta selata tapahtumia, sekä tehdä monipuolisia hakuja vanhoihin jo kuitattuihin tapahtumiin.

Yksittäisen tapahtuman tarkempia tietoja voidaan myös tarkastella monitorin liittymällä. Tarkempien tietojen sivulla tapahtumasta näytetään myös sen alkuperäinen tekstimuoto. DX200 tapahtumien tapauksessa yksityiskohtasivulla näkyy myös tapahtumaan liittyvä korjausohje.

Tapahtumamonitori tarjoaa tapahtumakerääjille HTTP-rajapinnan tapahtumien kantaan tallennusta varten.

3.4.1 Sääntökone

Tapahtumamonitoriin kuuluu sääntökone (RuleEngine) joka toisaalta on oma kokonaisuutensa mutta kuitenkin niin sidoksissa monitorisovellukseen että sen kuvaus liitetään tähän yhteyteen.

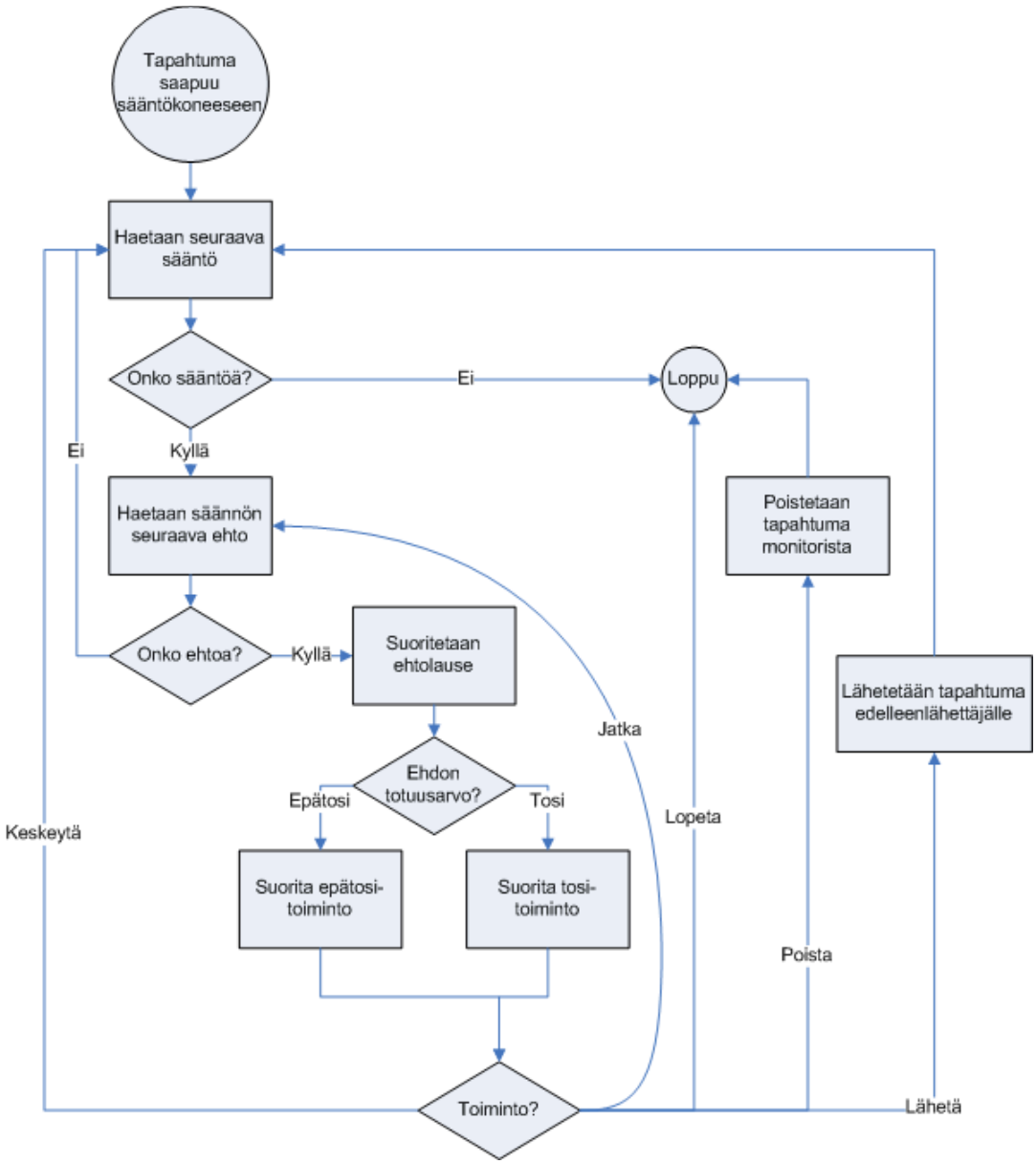
Sääntökone käynnistetään jokaisella kerralla kun jokin kerääjäsovellus ottaa yhteyden HTTP-rajapintaan ja pyytää tapahtuman kantaan tallennusta. Viime kädessä sääntökoneeseen asetetut säännöt päättävät tallennetaanko tapahtuman tiedot kantaan, hylätäänkö tapahtuma vai lähetetäänkö tapahtuma eteenpäin.

Säännöt päättävät myös sen mitkä tapahtumat lähetetään eteenpäin ja mille lähetyslistalle tapahtumat välitetään. Edelleen lähetettävän tapahtuman sääntökone tallentaa tapahtumalähettäjän tietokantatauluun lähetettäväksi, mukana lähetysajat ja lähetysreitit (tekstiviesti, sähköposti, puhelinsoitto).

Toiminto	***Kuvaus***
Jatka	Jatkaa sääntöketjun läpikäyntiä seuraavasta ehdosta.
Poista	Estää tapahtuman tallentumisen tapahtumakantaan ja keskeyttää sääntöketjun läpikäynnin.
Keskeytä	Keskeytetään säännön ehtojen läpikäynti ja otetaan seuraava sääntö käsittelyyn.
Lopeta	Lopettaa sääntöketjun läpikäynnin.
Lähetä	Lähettää tapahtuman määritellylle lähetyslistalle ja keskeyttää sääntöketjun läpikäynnin.

Taulukko 1: Sääntöjen toiminnot

Kuvassa 2 on esitetty sääntökoneen toiminta vuokaavion avulla. Sääntökone käynnistetään kun tapahtuma saapuu kantaan. Sääntökone hakee määritetyt säännöt järjestyksessä tietokannasta ja vertaa tapahtuman tietoja yksitellen jokaista sääntöä ja jokaisen säännön ehtoa vasten. Kaikkiin säännön ehtoihin määritellään myös toiminto tosi- ja epätosi-tilanteita varten joista toinen suoritetaan aina kun ehtoa käsitellään. Taulukossa 1 on lyhyesti selitettynä kaikki mahdolliset toiminnot.



Kuva 2: Sääntökoneen toiminta

3.5 Tapahtumalähettäjä

Tapahtumamonitorin sääntökone tallentaa tapahtuman tapahtumalähettäjän tietokantatauluun, jos jonkin ehdon ajettavaksi toiminnoksi on merkitty “Lähetä”. Tapahtumalähettäjä lähettää sääntökoneen välittämiä tapahtumia eteenpäin ehdossa määrätylle listalle. Lähetyslistoja voidaan ylläpitää monitorisovelluksen käyttöliittymän kautta samalla tavoin kuten sääntöjäkin.

Lähetyslistalle voidaan määritellä useita kohteita – kaikkiin määritellään myös lähetysviive, aika joka odotellaan ennen tapahtuman lähetystä – joihin tapahtuma lähetetään. Kohteita on tällä hetkellä kolme kappaletta: Tekstiviesti, sähköposti ja puhelinsoitto.

Jos jokin tapahtuma käynnistää säännöissä ”Lähetä”-toiminnon, lähetetään tapahtuman eteenpäin riippuen lähetyslistan kohteista. Sähköposti- ja tekstiviestilähetyksissä välitetään tapahtuman tiedot mukana, sähköpostissa kaikki tiedot ja tekstiviestissä tiivistelmä. Tekstiviestin tapahtumalähettäjä välittää SMS-Gateway sovellukselle, joka tämän järjestelmän tapauksessa on Radiolinjan toimittama.

Puhelinsoiton tapauksessa tapahtumalähettäjä valitsee kohdenumeron modeemilla ja soittaa siihen 30 sekunnin ajan, jonka jälkeen tapahtumalähettäjä katkaisee yhteyden. Tapahtumalähettäjä ei välitä vastataanko puhelinsoittoon vai ei. Puhelinsoittoon tapahtumalähettäjä käyttää yksinkertaista sarjaporttiin kytkettävää modeemia ja Perl-ohjelmointikielen Device::Modem-modulia.

4 JÄRJESTELMÄN RAKENNE

Kappaleessa kuvataan järjestelmän jokaisen osasovelluksen luokkarakenne. Tietokantarakenteet esitellään myös jokaiselle osasovellukselle paitsi kerääjälle jolla kantarakennetta ei ole. Osasovelluksia esiteltäessä käytetään esimerkkitapahtumina DX 200 -laitteen tapahtumalistausta (ks. 3.2).

4.1 Tapahtumakerääjä

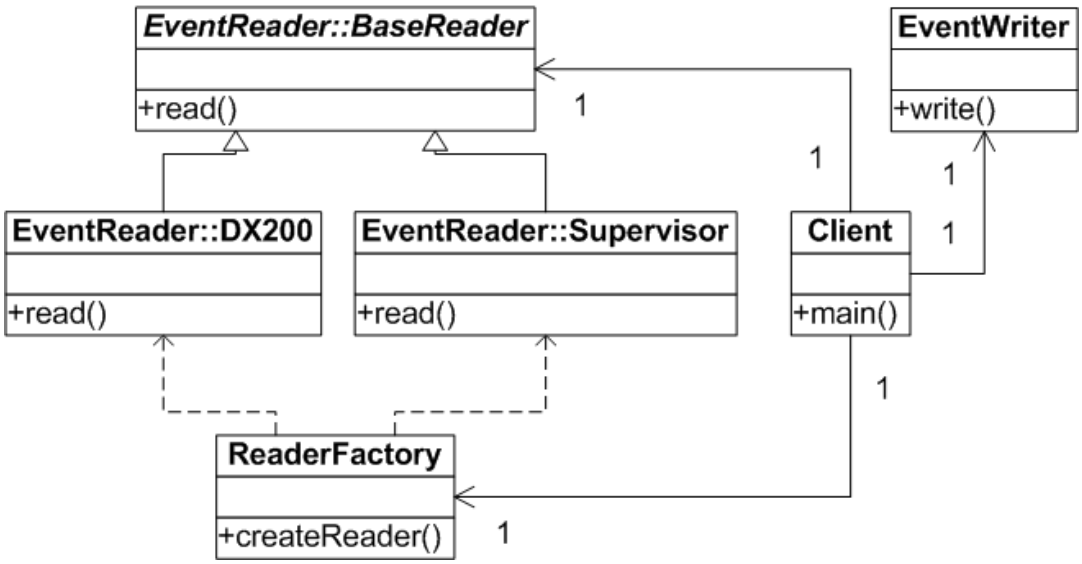
Tapahtumakerääjiä on monille eri lähdelaitteille ja kerääjät käyttävät yhteistä HTTP-rajapintaa tapahtumien tallentamiseen. Useilta lähdelaitteilta tulee kerääjäsovellukselle myös kuittausviestejä aikaisemmin tallennettuihin tapahtumiin. Kuittausviestien tallentaminen hoidetaan samantapaisella HTTP-rajapintakutsulla kuin tavallisen tapahtuman tallennus.

Yksi ajossa oleva tapahtumakerääjä muodostuu kahdesta erillisestä luokasta sekä luokat käynnistävästä pääsovelluksesta. Käynnistettävät luokat ovat lukija (EventReader::Base-luokasta peritty aliluokka) ja kirjoittaja (EventWriter). Näistä kahdesta lukija kerää tapahtumatietoja – seuraamalla lokitiedostoja tai tarkkailemalla telnet-yhteyttä – muuntaa tiedot XML-muotoon ja tallentaa kovalevylle konfiguraatiotiedostossa määriteltyyn spool-hakemistoon. Kirjoittaja tarkkailee samaa hakemistoa, hakee lukijan keräämät tapahtumat levyltä ja lähettää HTTP-rajapinnan kautta tapahtumat tapahtumamonitorille tallennettavaksi tapahtumakantaan. Kaikille kerääjäsovelluksille kirjoittaja-luokka on samanlainen oli lähdelaitteena mikä tahansa. Lukija-luokka toimii ns. Plugin-periaatteella, eli jokaiselle lähteelle kirjoitetaan oma toteutuksensa joka sitten muuntaa lähteen tapahtumat parhaimmalla mahdollisella tavalla kirjoittaja-luokan ymmärtämään muotoon. Lukija on aina EventReader::Base-luokan aliluokka, joka periyttämällä erikoistaa tapahtumien lukemiseen käytetyn read-metodin.

Kerääjän pääsovellus – joka käynnistää lukija- sekä kirjoittajasäikeet – saa

parametrinaan konfiguraatiotiedoston jossa kerrotaan käynnistettävän lukija-luokan nimi. Pääohjelma lataa dynaamisesti levyltä lukijan moduulin, luo olion ja käynnistää omaan säikeeseen lukijan read-metodin. Pääohjelma käynnistää myös kirjoittajan tarkkailemaan spool-hakemistoa.

Kerääjä suunniteltaessa päätettiin lukijan käynnistykseen käyttää ”Abstrakti tehdas”-suunnittelumallia (Abstract Factory) /4,6/. Mallissa tehdas-luokka tuottaa dynaamisesti, sovelluksen ajon aikana saadun luokan nimen perusteella ilmentymän luokasta luomismetodin kutsulla. Metodin nimi on lukijoiden tapauksessa createReader ja tehdas on ReaderFactory. Kuvassa 3 on esitetty kerääjän luokkakaavio, jossa pääsovelluksen virkaa toimittaa Client-metaluokka.



Kuva 3: Kerääjän luokkakaavio

4.2 Tapahtumamonitori

Tapahtumamonitori on toteutettu Apprunner-sovelluspalvelin ratkaisuna, tavallisena www-palveluna. Käyttöliittymä koostuu joukosta Apprunner-kuvauskielisiä sivuja, joista muodostetaan dynaamisesti HTML-koodattuja sivuja www-selaimella käyttöä varten.

Tapahtumamonitorilla on yksi tietokantataulu nimeltä event, josta AppRunner-sovelluspalvelin noutaa tapahtumien tietoja. Taulukossa 2 on lueteltu tietokannan kentät ja niiden kuvaukset. Taulussa on useita kenttiä joiden arvot voidaan jättää tyhjäksi, koska kaikkien järjestelmien tapahtumissa ei luonnollisesti ole kaikkia samoja tietoja kuin esimerkiksi DX200 järjestelmässä. Vain muutamat kentät ovat pakollisia ja ne onkin merkitty taulukossa tähdellä. Kaikki ei pakolliset kentät tapahtumakerääjät voivat jättää tarpeen tullen tyhjiksi, jolloin ne eivät myöskään näy monitorin käyttöliittymässä millään tavalla. Jos jonkin tapahtumalähteen tapahtumista puuttuu pakollisia kenttiä jää tapahtumakerääjän tehtäväksi generoida tapahtumaan järkevät tiedot, siten että tapahtuman tallennus kantaan on mahdollinen.

Taulussa on myös varattu yksi kenttä tapahtuman XML-esitykselle, johon tallennetaan järjestelmään sopimattomat tiedot ja myös tapahtuman alkuperäinen esitys. XML-esityksen avulla voidaan tapahtuman yksityiskohtasivulla esittää myös nämä tauluun sopimattomat tiedot, jolloin tapahtumista ei jää mitään huomiotta.

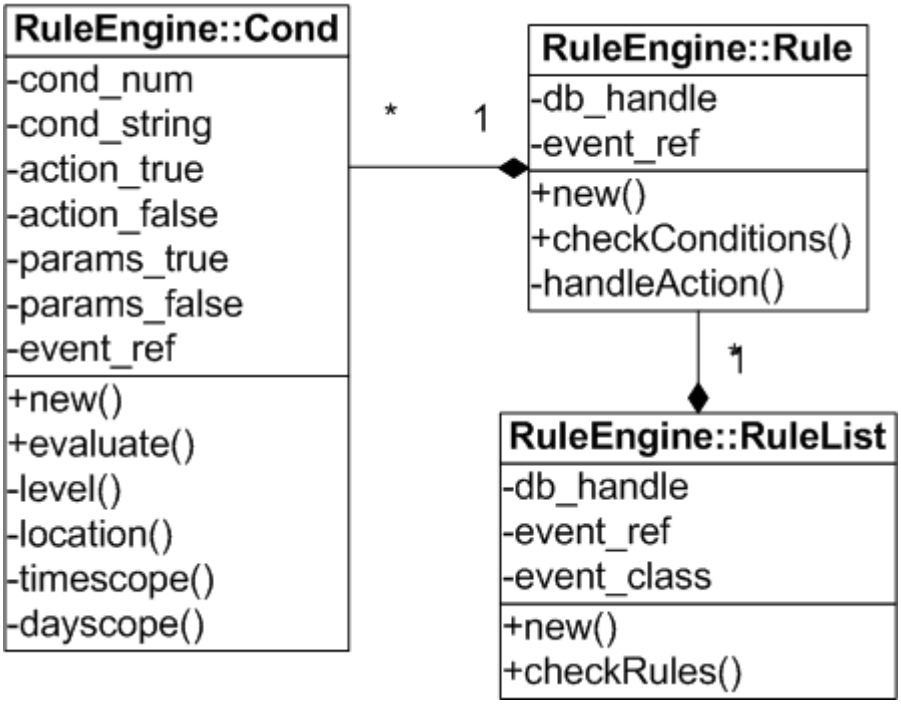
Kentän nimi	***Tyyppi***	***Kuvaus***
id	tnteger	Tietokantarivin yksilöivä id-arvo.
timestamp	timestamp	Tapahtuman saapumisaika.
timestamp_ack	timestamp	Tapahtuman kuittausaika.
timestamp_taken	timestamp	Aika, jolloin tapahtuma on otettu käsittelyyn.
timestamp_open	timestamp	Aika, jolloin tapahtuma on merkitty pois käsittelystä.
event_type*	integer	Tapahtuman tyyppi. (Ilmoitus, Hälytys ja Häiriö)
event_id	Integer	Tapahtuman sekvenssinumero. Laitteelta tuleva yksilöivä numero, käytetään kuittauksessa.
event_level*	integer	Tapahtuman taso.
event_xml*	text	Tapahtuman XML-esitys.
event_state*	Integer	Tapahtuman tila. (Avoin, Käsittelyssä ja Kuitattu)
event_class*	integer	Tapahtuman luokka.
event_location*	text	Tapahtuman sijainti, esim. maantieteellinen.
event_desc*	text	Tapahtuman kuvaus.
event_code	integer	Tapahtuman koodi.

Taulukko 2: Event-taulun kenttäkuvaukset

4.2.1 Sääntökone

Sääntökone koostuu kolmesta luokasta. Sääntölista-luokasta (RuleList), sääntö-luokasta (Rule) sekä ehto-luokasta (Cond).

Kuvassa 4 on sääntökoneen luokkakaavio. Yhdessä sääntölistassa voi olla useita sääntöjä ja yhteen sääntöön voi kuulua useita ehtoja.



Kuva 4: Sääntökoneen luokkakaavio

Monitori käynnistää sääntökoneen kun tapahtumamonitorin kantaan kirjoitetaan HTTP-rajapinnan kautta uusi tapahtuma. Sääntökone käynnistetään luomalla uusi ilmentymä RuleEngine::RuleList-luokasta ja kutsumalla sen checkRules-metodia. RuleList-luokka hakee muodostuessaan tietokannasta kaikki tapahtuman luokkaan kuuluvat säännöt ja muodostaa kaikista löydetyistä säännöistä Rule-luokan ilmentymän jotka sijoitetaan rules-taulukkoon. Säännöt puolestaan muodostavat cond_list-nimisen taulukon Cond-luokan ilmentymistä.

Kun kaikki säännöt ja ehdot on luettu taulukoihin, käynnistetään sääntöjen läpikäynti checkRules-metodilla. Metodi käy läpi sääntöjä yksi kerrallaan ja kutsuu jokaisen säännön kohdalla checkConditions-metodia, joka käy läpi sääntöön liittyvät ehdot. Metodi käsittelee säännön ehdot yksi kerrallaan ja kutsuu jokaiselle ehdolle evaluate-metodia, joka selvittää ehdon totuusarvon. Totuusarvon selvityksen jälkeen checkConditions suorittaa ehdoissa määriteltyjä toimintoja riippuen ehdon totuusarvosta handleAction-metodilla. Jokaiselle ehdolle määritellään tosi- ja epätosi-toiminto, jolloin jokaiselle ehdolle ajetaan ainakin jokin toiminto. Mahdolliset toiminnot on listattuna taulukossa 1, sivulla 10.

Kun ehdon suoritettavana toimintona on ”Lähetä” RuleList tallentaa tapahtuman tiedot tapahtumalähettäjän tietokantatauluun (Forw_spool) ja lähettää tapahtumalähettäjälle signaalin, että uusi tapahtuma on saapunut lähetyskantaan.

Sääntökoneella on kaksi tietokantataulua, joista toiseen (Rule) tallennetaan säännöt, ja toiseen (Condition) tallennetaan sääntöön liittyvät ehdot. Taulukoissa 3 ja 4 on taulujen kentät sekä niiden kuvaukset.

Kentän nimi	***Kuvaus***
id	Tietokantarivin yksilöivä id-numero.
rule_num	Säännön järjestysnumero.
event_class	Kertoo mihin tapahtumaluokkaan sääntö vaikuttaa.
active	Säännön aktiivisuus.

Taulukko 3: Rule-taulun kentät

Kentän nimi	***Kuvaus***
id	Tietokantarivin yksilöivä id-numero.
rule_id	Säännön id, johon ehto kuuluu.
cond_num	Ehdon järjestysnumero.
cond_string	Ehdon esitys tekstimuodossa.
action_true	Toiminto, joka suoritetaan kun ehto on tosi.
action_false	Toiminto, joka suoritetaan kun ehto on epätosi.
active	Ehdon aktiivisuus.
params_true	Parametrit jotka välitetään suoritettavalle toiminnolle ehdon ollessa tosi. Tällä hetkellä vain lähetä-toiminto käyttää tätä.
params_false	Parametrit jotka välitetään suoritettavalle toiminnolle ehdon ollessa epätosi.

Taulukko 4: Condition-taulun kentät

4.3 Tapahtumalähettäjä

Tapahtumalähettäjä rakentuu ns. Backend-osasta ja varsinaisista tapahtuman lähettäjistä. Jokaiselle lähetysreitille on oma lähettäjänsä kun taas backend-osia on vain yksi kaikkia lähetysreittejä varten.

Backend odottaa signaalia sääntökoneelta ja sen saatuaan lukee Forw_spool-taulusta lähetettävän tapahtuman tiedot ja Forw_list_target-taulusta lähetyskohteen ja reitin. Backend laskee tapahtumalle lähetysajan ja päivittää lähetysajan, lähetyskohteen ja reitin Forw_spool-tauluun.

Lähettäjät valvovat Forw_spool-taulua ja havaitessaan itselleen kuuluvan lähetettävän tapahtuman tarkastavat onko aika lähettää tapahtuma eteenpäin. Lähetyksen jälkeen lähettäjä poistaa viittauksen tapahtumaan Forw_spool-taulusta. Forw_spool-taulun kenttien kuvaukset löytyvät taulukosta 7.

Kentän nimi	***Tyyppi***	***Kuvaus***
id	integer	Tietokantarivin yksilöivä id-numero.
name	text	Lähetyslistan nimi.

Taulukko 5: Forw_list-taulun kentät

Tapahtumamonitorin käyttöliittymän kautta Forw_list-tauluun tallennetaan lähetyslistoja ja Forw_list_target-tauluun lähetyslistojen kohteita. Yhteen lähetyslistaan voi kuulua useita kohteita. Tallennettaessa kohteita, käyttäjä voi valita lähetysreitin, lähetysviiveen ja kohdeosoitteen johon tapahtuma lähetetään. Taulukoissa 5 ja 6 on lähetyslistoihin liittyvät taulut ja niiden kenttien kuvaukset.

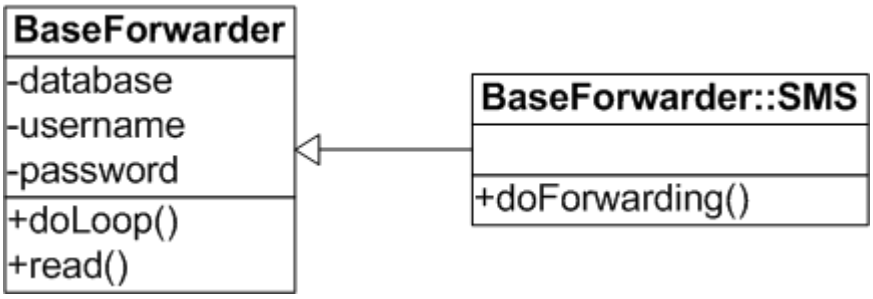
Kentän nimi	***Tyyppi***	***Kuvaus***
id	integer	Tietokantarivin yksilöivä id-numero.
list_id	integer	Lähetyslistan nimi.
route	integer	Lähetysreitti. (SMS, Sähköposti jne.)
delay	integer	Lähetysviive sekunteina.
target	text	Kohdeosoite/numero.

Taulukko 6: Forw_list_target-taulun kentät

Kentän nimi	***Tyyppi***	***Kuvaus***
id	integer	Tietokantarivin yksilöivä id-numero.
event_id	integer	Lähetettävän tapahtuman id.
list_id	integer	Lähetyslistan id.
deliver_at	timestamp	Laskettu lähetysaika.
route	integer	Lähetysreitti.
target	text	Kohdeosoite/numero.
event_xml	text	Tapahtuman XML-esitys.

Taulukko 7: Forw_spool-taulun kentät

Lähettäjät muodostuvat periyttämällä BaseForwarder-yliluokan. Yliluokalla on doLoop-metodi, jota aliluokka kutsuu käynnistyessään. doLoop kutsuu yliluokan read-metodia, jolla lähettäjät lukevat itselleen kuuluvat lähetettävät tapahtumat Forw_spool-taulusta. Metodin kutsun jälkeen doLoop kutsuu aliluokkien toteuttamaa doForwarding-metodia, jossa varsinainen lähetys tapahtuu. Tapahtuman lähetyksen jälkeen lähettäjät poistavat lähetykseen viittaavan rivin Forw_spool-taulusta. Kuvassa 5 on lähettäjän luokkakaavio.



Kuva 5: Tapahtumalähettäjän luokkakaavio

5 TULOKSET JA TULEVAISUUS

Insinöörityön tuloksena on tuotettu järjestelmä jolla voidaan valvoa ja käsitellä erilaisissa laitteissa ja sovelluksissa muodostuvia tapahtumia. Käyttökokemuksia järjestelmästä on vasta vähän, koska se on osittain kehityksen alla tätä työtä kirjoitettaessa. Ensikokemuksien perusteella käyttöliittymä toimii kuitenkin hyvin ja säännötkin toimivat odotetulla tavalla.

Käyttöliittymässä olisi kuitenkin parannettavaa, esimerkiksi sääntöjen käsitteleminen on kömpelöä ja käytettävyys paranisi huomattavasti jos sääntöjen määritteleminen saataisiin hoidettua yhden sivun latauksella, eikä käyttäjää pakotettaisi kulkemaan usealla sivulla saadakseen luotua järkevän säännön. Muutos vaatisi luultavasti Javascriptin käyttöä ja se tuottaa jälleen omat ongelmansa, kuten selainriippuvuudet ja myös mahdolliset käyttöjärjestelmäriippuvuudet.

Tapahtumakerääjien kanssa on ollut joitakin ongelmia, mutta kaikki ongelmat ovat liittyneet lähdelaitteen tapahtumatietojen epämääräiseen ennalta-arvaamattomaan muotoiluun tai muuhun pikkuseikkaan, eivätkä ole pahemmin häirinneet järjestelmän testikäyttöä. Suurin ongelma järjestelmää toteutettaessa – varsinkin tapahtumakerääjiä – on ollut alusta loppuun asti se etteivät lähdelaitteiden valmistajat ole kunnolla standardoineet tai edes dokumentoineet laitteidensa tai sovellustensa tapahtumatietojen esitysmuotoa.

Jos yhdellä koneella on käynnissä useampi tapahtumakerääjä, käynnistetään jokaista kerääjää kohti yksi kirjoittaja- ja yksi lukija-säie. Kaikille kerääjille riittäisi yksi kirjoittaja-säie ja useita lukija-säikeitä jokaista keräyslähdettä varten. Nykyisen käytännön vuoksi tarvitaan kerääjille erillinen käynnistys-skripti joka käynnistää tarvittavat tapahtumakerääjät, vaikka käytännössä kerääjäsovellus voisi itse hoitaa kaikkien tarvittavien säikeiden käynnistyksen.

Tapahtumalähettäjän puhelinsoitto ominaisuus on melko merkityksetön, koska sillä ei saada välitettyä varsinaista tietoa tapahtumasta vastaanottajalle. Jatkokehitys-ideana voisikin miettiä miten vaikeata olisi liittää puhelinsoittoon puhesyn-tetisaattori joka voisi lukea tapahtuman otsikkotiedon tai kuvaustiedon puhelimeen. Ominaisuus on kuitenkin vanhassa muodossaankin siinä mielessä hyödyllinen että

jos sähköpostit ja/tai tekstiviestit ovat päässeet lipsahtamaan valvojan tarkan silmän alta voi viimeistään puhelinsoitto herättää tarkastamaan saapuneet tapahtumat ja laitteiden tilan.

Järjestelmästä haluttiin alun perin mahdollisimman yleiskäyttöinen, jolloin siihen olisi mahdollisimman helppo liittää uusia laitteita valvonnan alle. Tässä tavoitteessa onnistuttiin hyvin ja nykyiseen järjestelmään uuden kerääjän lisääminen on suhteellisen yksinkertaista. Monitorin tietokannassa on kuitenkin monia kenttiä joihin kaikista laitteista ei välttämättä saada täytettyä mitään tietoa – eikä tiedon generointikaan tunnu järkevältä – josta syystä kantaan tallennettavien kenttien määrän pitäisi vähentyä ja tallentaa suurin osa tapahtumatiedoista XML-muodossa.

Kaiken kaikkiaan järjestelmä oli menestys ja lyhyiden käyttökokemuksien perusteella valvojien työskentely on helpottunut ja ongelmatilanteisiin kuluva reagointiaika on lyhentynyt.

LÄHTEET

1 Kulma, Teemu, Opinnäytetyö, Tampereen ammattikorkeakoulu Sähkötekniikan osasto 2004, 27s + 4 liites.

2 Etelä-Satakunnan Puhelin Oy:n [www-sivu]. [viitattu 15.1.2005] Saatavissa: http://www.esp.fi

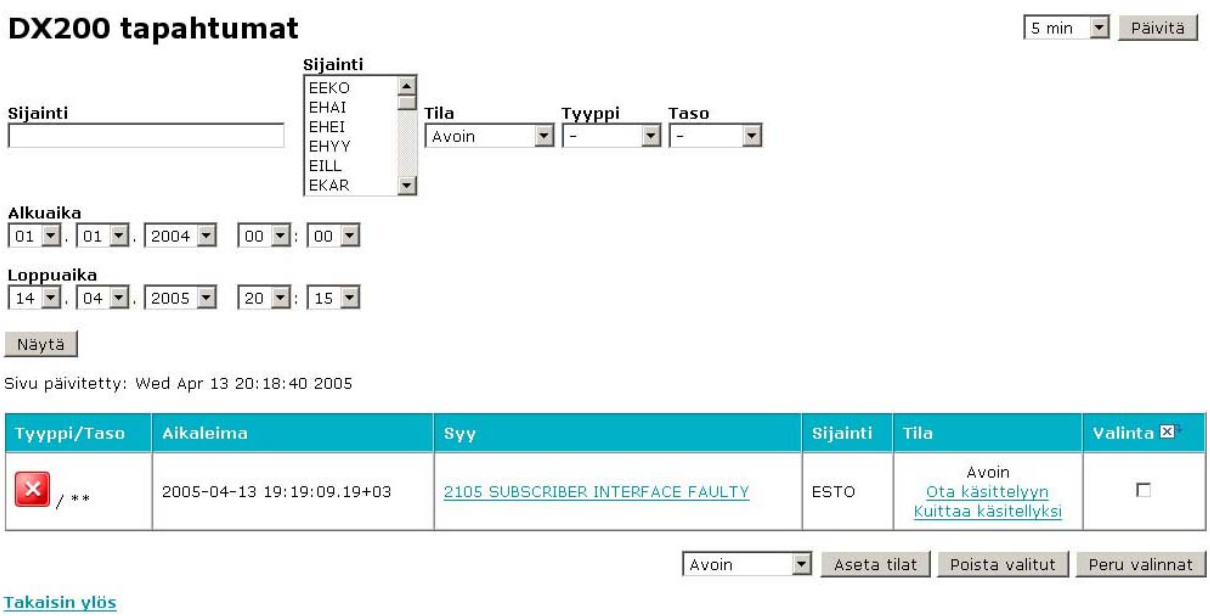
3 Netland Oy [www-sivu]. [viitattu 15.1.2005] Saatavissa: http://www.netland.fi

4 Perl Design Patterns, Part 3 [www-sivu]. [viitattu 10.2.2005] Saatavissa: http://www.perl.com/pub/a/2003/08/15/design3.html

5 Larry Wall, Tom Christiansen, Jon Orwant, Programming Perl (3rd Edition). O'Reilly 2000. 1092 s.

6 Erich Gamma, Richard Helm, Ralph Johnson, John Vissides, Design Patterns : elements of reusable object-oriented software (1st edition). Addison-Wesley 1995. 395 s.

LIITTEET



Tapahtumamonitorin etusivu

Tapahtuma: SUBSCRIBER INTERFACE FAULTY (2105)

Tapahtuman yhteinen data	
ID	320550
Tapahtuman tunniste	6278
Kuvaus	SUBSCRIBER INTERFACE FAULTY
Aikaleima	2005-04-14 16:00:29.14+03
Luokka	DX200
Sijainti	ERS2
Koodi	2105
Taso	**
Tyyppi	Hälytys
Tila	Avoin Ota käsittelyyn Kuittaa käsitellyksi Ajassa: 2005-04-14 15:52:37.581467+03

Tapahtuman yksilöllinen data	
Keskus	DX220-EPI2
Laitetyyppi	SWITCH
Hex-tarkenne	LIA10 8d 01
Tapahtuman data	DX220-EPI2 ERS2 SSU-1-1 SWITCH 2005-04-14 16:00:29.14 ** ALARM SUB-1-81 IB001-37 PXRECE (6278) 2105 SUBSCRIBER INTERFACE FAULTY LIA10 8d 01

Tapahtuman ohje

Merkitys

Järjestelmän jatkuva valvonta on havainnut analogisen tilaajaliitännän vialliseksi. Vian syynä on vaurioitunut tilaajaliitäntäpistoyksikkö. Vian johdosta yhteyden laatu voi heiketä tai liikenne voi estyä kokonaan ko. tilaajalla.

Lisätietokentät

1 tilaajaliitäntäpistoyksikön tyyppi

2 tilaajaliitäntäpistoyksikön indeksi

3 tilaajan paikka pistoyksiköllä

Toimintaohje

Suorita tilaajaliitännän mittaukset (syöttöjännite, vaimennus, toimintatesti, tilaajasilmukan tilatesti ja silmukkavalinnan vastaanottotesti) kyseiselle tilaajaliittymälle vian paikallistamiseksi esim. MML-komennolla:

2ULI:LP=x-y-z;

missä
LP = looginen paikkanumero
x = porras
y = moduuli
z = liittymän numero moduulissa.

Lisätietoja:
Tilaajajohtojen testaus, Komentokäsikirja.

Jos liitännässä havaitaan vika (mittaustuloksen viereen tulostuu '*'), vaihda viallinen liitäntäpistoyksikkö, ks. Suoritusohjeet pistoyksikön vaihtamiseksi, Kunnossapitokäsikirja.

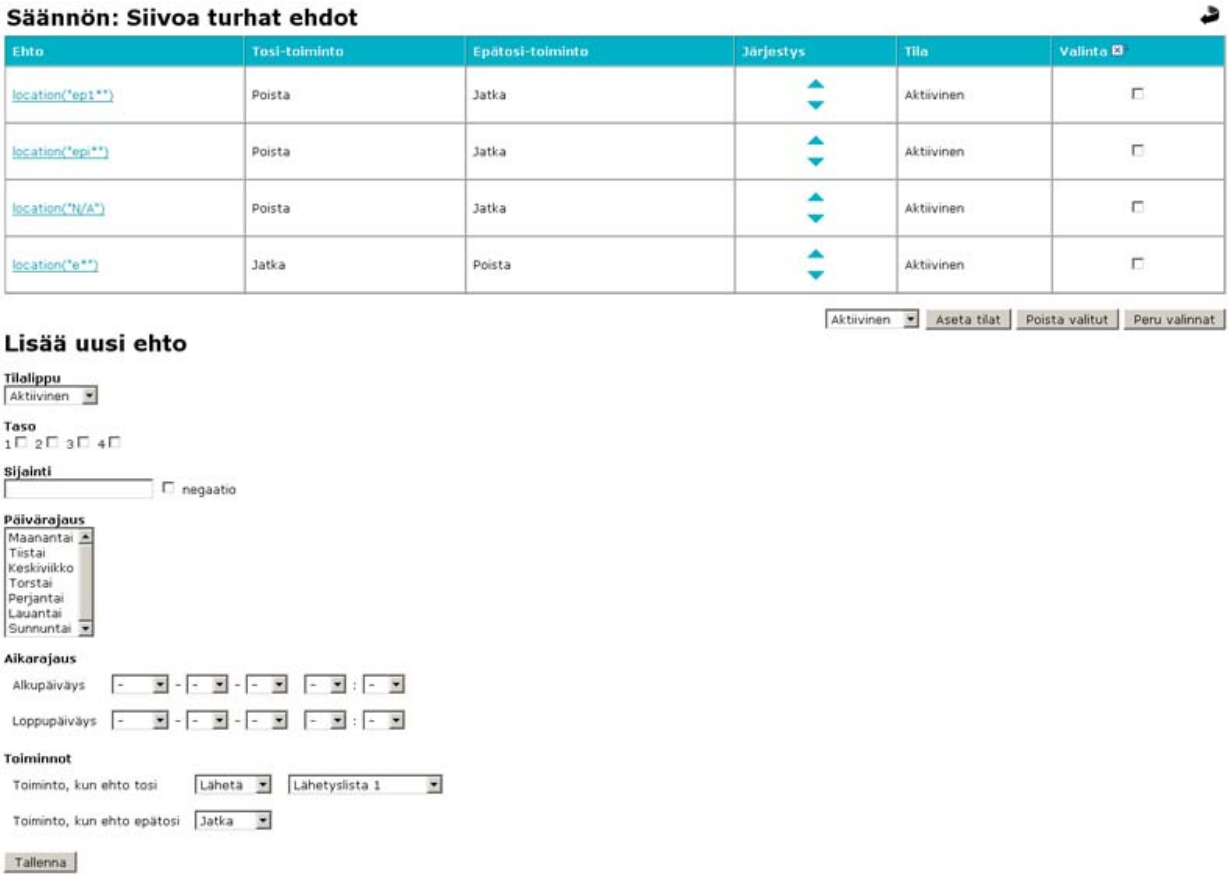
Peruuttaminen

Älä peruuta hälytystä. Järjestelmä peruuttaa hälytyksen automaattisesti kun hälytyksen aiheuttanut vika on korjattu.

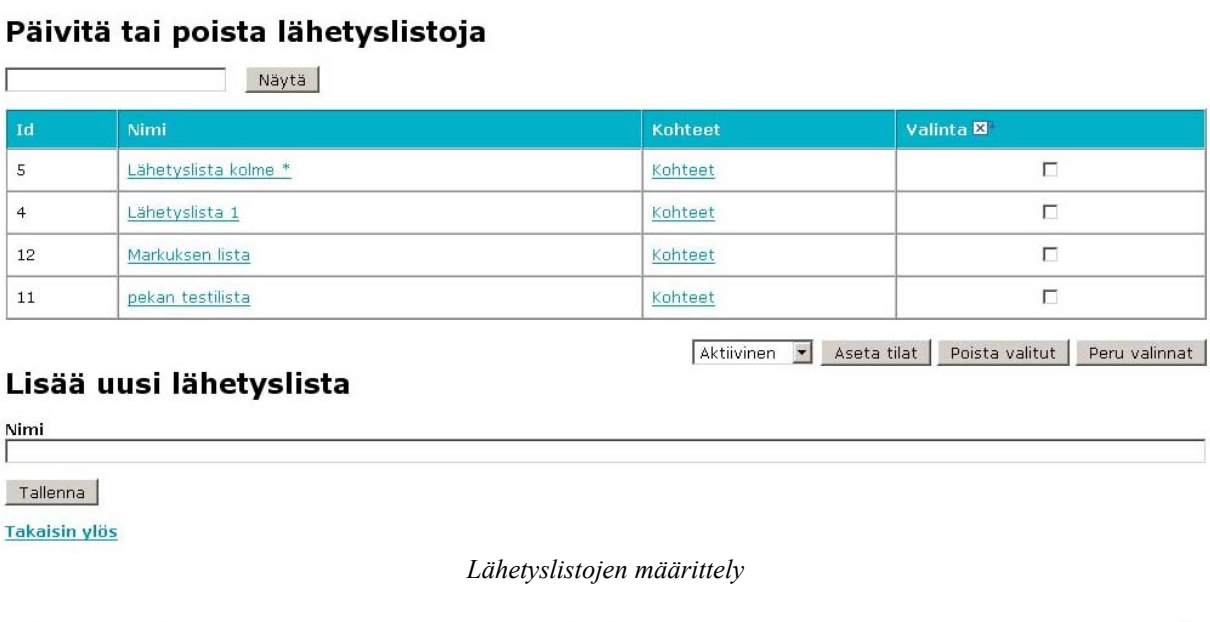
Takaisin ylös

Tapahtuman tarkemmat tiedot

Sääntöjen määrittely



Säännön ehtojen määrittely



Lähetyslistojen määrittely

Lähetyslistan kohteiden määrittely

```
#########################################################
#FILE: ReaderFactory.pm
#########################################################
package ReaderFactory;

use lib '/usr/local/event_monitor/EventCollector/';
use lib '.';

use strict;

sub createReader
{
    my ($self, %params) = @_;

    ($params{reader}->{class}) = ($params{reader}->{class} =~ m|^(\w+)$|);
    ($params{dxhost})          = ($params{dxhost} =~ m|^([\w\.\-]+)$|);
    ($params{dxport})          = ($params{dxport} =~ m|^(\w+)$|);

    my ($requested_type) = ($params{reader}->{class} =~ m|^(\w+)$|);

    my $location = "EventReader/$requested_type.pm";
    my $class = "EventReader::$requested_type";

    require $location;

    return $class->new(%params);
}

1;
```