Hariton Tuukkanen jr

Palohälytys-ohjelmistolaajennus EIRIS-alustaan

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
17.04.2011

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Hariton Tuukkanen jr Palohälytys-ohjelmistolaajennus EIRIS-alustaan 27 sivua 17.4.2011
Tutkinto	Insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	-
Ohjaajat	yliopettaja Antti Piironen toimitusjohtaja Jorma Salonen

Tässä insinöörityössä kerrotaan Telepulssi Oy:ssä suunnitellusta ja toteutetusta Palohälytys-ohjelmistolaajennuksesta Visonic Technologies ltd:n EIRIS-ohjelmistoon. Ohjelmiston tarkoituksena on välittää palokeskusten hälytysviestit EIRIS-ohjelmiston avulla rakennuksissa oleviin käytävänäyttöihin. Hälytystilanteessa käytävänäyttöihin ilmestyy tulipalon sijaintiin liittyvää tietoa. Palohälytyskeskuksena toimii Ab Hedengren oy:n Prodex-paloilmoitinkeskus.

Uusien paloturvallisuusmääräysten myötä oli Meilahden yhteispäivystyssairaalaan vaadittu lisättävän paloturvallisuutta. Lisätystä turvallisuudesta oli useita eri vaihtoehtoja. Vaihtoehtoina oli muun muassa koko talon kattava sprinklerijärjestelmä tai tarkennettu palohälytystieto rakennuksen henkilökunnalle ja sen asiakkaille. Silloiselta työnantajaltani Telepulssi Oy:ltä pyydettiin vaihtoehtoista ratkaisua tarkennettuun palohälytystietojärjestelmään.

EIRIS-ohjelmisto on hyvin monipuolinen työkalu yritysten turvajärjestelmien keskeiseen hallintaan. Ohjelmiston tarkoituksena on liittää kaikki yrityksen turvajärjestelmät yhden käyttöliittymän alle.

Insinöörityössä kehitetyllä Palohälytys-ohjelmistolla on tarkoitus liittää Prodex-merkkiset palohälytyskeskukset EIRIS-ohjelmistoon. Ohjelmisto mahdollistaa saapuvien palohälytysten edelleen välityksen loppukäyttäjille eri metodein. Palohälytys-ohjelmistolaajennuksen kehityksessä päädyttiin käyttämään prototyyppi -ohjelmointimallia. Ohjelmointimalli mahdollisti EIRIS-ohjelmistolaajennuksen nopean kehitystyön. Prototyyppi-ohjelmointimallin ansiosta ohjelmistolaajennuksessa suunniteltujen toiminnallisuuksien lopputulokset oli mahdollista nähdä heti. Ennen Palohälytys-ohjelmistolaajennuksen lopullista versiota piti useat toiminnallisuudet suunnitella ja rakentaa uudelleen.

Insinöörityössä saatiin aikaiseksi kaksi EIRIS-objektia: Palokeskus-objekti ja Ryhmienhallinta-objekti. Nämä aikaan saadut objektit mahdollistivat Prodex-palokeskusten liittämisen EIRIS-ohjelmistoon.

Avainsanat	ESPA 4.4.4 protokola, EIRIS, Prodex, ohjelmistolaajennus

Metropolia

Abstract

Authors Title Number of Pages Date	Hariton Tuukkanen jr A fire alarm program extension for EIRIS platform 27 pages 17 April 2011
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	-
Instructors	Antti Piironen, Senior Lecturer Jorma Salonen, Chief Executive Officer

This thesis is about a fire alarm program extension that was created by Telepulssi Oy for Visonic Technologies Ltd's EIRIS software. The purpose of the program extension was to send firealarm data to floor displays located inside the building. During an active firealarm the floor displays will show location related information about the active firealarm whereby text "FIRE ALARM" will appear on the displays. The Prodex firealarm center used in this project was created by a company called Hedengren Ltd.

Because of new fire regulations, the general emergency hospital in Meilahti. The hospital was required to provide additional fire safety facilities to the building. There were multiple options to consider from sprinkler systems to more detailed fire alarm location information for use building's personnel. Telepulssi Ltd, the author's employer at the time, was asked to create a detailed location information system.

EIRIS software is a versatile tool to manage a company's security systems. The purpose of the software is to bring all the companys security systems under on one user interface.

The purpose of the fire alarm software extension was to make it possible for EIRIS software to receive the firealarms from Prodex firealarm centers. This firealarm data imported to EIRIS software can be forwarded to the end users by different methods.

During the development of the fire alarm software extension a prototype programming mode was opted for. This programming model made the fast development of the software extension possible. The possibility to quickly view the end result of small prototypes was possible with this programming model. Before the final version of the software extension most of the features included in it had to be rebuilt and redesigned several times.

This thesis in creating two EIRIS objects: an object for fire alarm center and an object for group management. Using these two EIRIS objects it is possible to receive fire alarms from Prodex fire alarm centers.

Keywords	ESPA 4.4.4 protocol, EIRIS, Prodex, software extension

Sisällys

1 Johdanto

Paloilmoitinjärjestelmät ovat pakollinen osa rakennustekniikkaa jo pelkästään turvallisuuden lisäämiseksi ja henkilövahinkojen välttämiseksi. Melkein jokaisesta kodista löytyy paloilmoitinlaitteita, ovat ne sitten paristoilla toimivia tai osana kiinteistön suurempaa hälytysjärjestelmää. Paloilmoitinjärjestelmät ovat osa jokapäiväistä elämäämme, vaikka emme ole aina tietoisia asiasta. Niin kaupassa, sairaalassa, kirjastossa tai vaikka työpaikalla on jokainen kohteista suojattu paloilmoitinjärjestelmällä.

Tässä insinöörityössä kerron Telepulssi Oy:ssä suunnitellusta ja toteutetusta Palohälytys-ohjelmistolaajennuksesta Visonic Technologies ltd:n EIRIS-ohjelmistoon, C++-ohjelmointikieltä käyttäen. Ohjelmiston tarkoituksena on välittää palokeskusten hälytysviestit EIRIS-ohjelmiston avulla rakennuksissa oleviin käytävänäyttöihin. Hälytystilanteessa käytävänäyttöihin ilmestyy tulipalon sijaintiin liittyvää tietoa ja lisäksi teksti "TULIPALO". Palohälytyskeskuksena toimii Prodex-paloilmoitinkeskus.

Uusien paloturvallisuusmääräysten myötä oli Meilahden yhteispäivystyssairaalaan vaadittu lisättävän paloturvallisuutta. Lisätystä turvallisuudesta oli useita eri vaihtoehtoja. Vaihtoehtoina oli muun muassa koko talon kattava sprinklerijärjestelmä tai tarkennettu palohälytystieto rakennuksen henkilökunnalle ja sen asiakkaille. Silloiselta työnantajaltani Telepulssi Oy:ltä pyydettiin vaihtoehtoista ratkaisua tarkennettuun palohälytystietojärjestelmään.

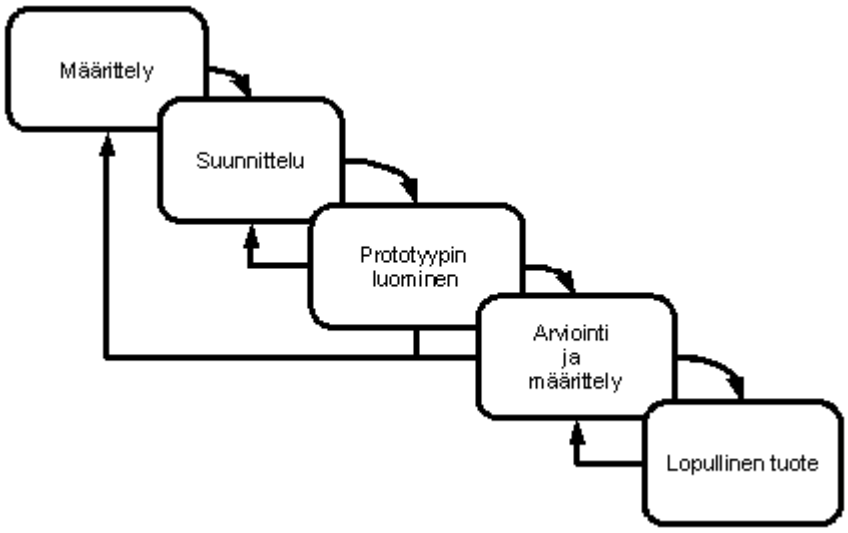
Toteutin ohjelmistolaajennuksen osin Telepulssi Oy:n toimitusjohtajan Jorma Salosen kanssa, osin itsenäisesti. Ohjelmistolaajennuksen määrittelyn ja testauksen osalta teimme aktiivista yhteistyötä. Suunnittelun ja toteutuksen osalta tein työtä hyvin itsenäisesti alusta loppuun.

Telepulssi Oy on perustettu vuonna 1978 ja se on erikoistunut hälytys-, viestintä- ja turvajärjestelmien toteuttamiseen. Telepulssi Oy on ollut useita vuosia mukana HUS:n sairaaloiden turvajärjestelmien toteuttamisessa. [6.]

Työskentelin Telepulssi Oy:ssä viisi vuotta. Tuona aikana toimenkuvani koostui muun muassa turvajärjestelmien suunnittelusta, kaapeloinnista, asennuksista, testauksista ja edelleen käytönopastuksista. Näin ollen vuosien varrella monien eri valmistajien turvajärjestelmien ”sielunelämä” on tullut hyvin tutuiksi. Näiden Telepulssissa opittujen sekä koulussa oppimieni tietojen pohjalta tämä insinöörityö on toteutettu.

2 Prototyyppi-ohjelmointimalli

Prototyyppi-ohjelmointimalli sopii hyvin tilanteisiin, joissa on hyvä nähdä etukäteen lopullisen tuotteen toimintamalli eli prototyyppi. Ohjelmointimallissa on seuraavat vaiheet: määrittely, suunnittelu, prototyypin luominen, arviointi ja määrittely sekä lopullinen tuote. Ohjelmointimalli on hyvin joustava. Alun määrittelyistä päästään nopeasti tekemään suunnittelutyötä ennen prototyypin luomista. Ohjelmointimallin määrittelyyn voidaan hypätä vaikka kesken prototyypin luomisen jos alkaa näyttää siltä, ettei työ etene kuten pitäisi. Kuvan 1 prototyyppiohjelmoinnin prosessikaaviokuva esittää kuinka kyseistä ohjelmointimallia pitäisi noudattaa. [2.]



Kuva 1. Prototyyppiohjelmoinnin prosessikaavio. [2.]

Uuden projektin alussa prototyyppejä saattaa kasaantua hyvinkin monia. Prototyyppejä on kahdenlaisia: hylättäviä ja kehittyviä prototyyppejä. [2.]

Hylättävät prototyypit, kuten nimikin sanoo, hylätään tuotantokäytöstä, mikäli prototyypin suunnitteluun on valittu väärä lähestymistapa. Tämä voi ilmetä esimerkiksi tapauksessa, jossa prototyyppi ei toimi määritellyllä tavalla. Tällaisessa tapauksessa suunnittelu on parasta aloittaa uudelleen puhtaalta pöydältä. [2.]

Kehittyviä prototyyppejä tullaan jalostamaan tuotantokäyttöön mahdollisesti useamman uuden prototyypin kautta. Näiden prototyyppien kehitys ja toiminnallisuus on alusta asti ollut oikeilla jalanjäljillä. [2.]

Lopullinen tuote voi ensimmäisten testien jälkeen useasti olla vain uusi prototyyppi. Lisää aiheesta kerrotaan luvussa 8.

3 ESPA-protokolla

ESPA 4.4.4 on sarjamuotoinen protokolla, joka lähettää lyhyitä viestejä erilaisten sarjaliikenteiden kautta kuten RS-232, RS-422 ja RS-485. ESPA-protokolla 4.4.4 sai syntynsä vuonna 1984 eurooppalaisten hakulaitevalmistajien toimesta. Protokollasta muodostui hyvin nopeasti teollisuuden standardi. [1.]

ESPA-protokollan yhdessä dataväylässä täytyy olla yksi aktiivinen päälaite, ja muut laitteet toimivat passiivisina oheislaitteina. Päälaite antaa jokaiselle oheislaitteelle mahdollisuuden lähettää tietoa dataväylälle. Jokaiselle dataväylään liitetylle laitteelle on annettu uniikki osoite. ESPA-protokollassa kommunikointi tapahtuu pelkästään laitteiden osoitteiden perusteella. [1.]

3.1 Kommunikointi

Päälaite lähettää järjestyksessä jokaiselle dataväylässä olevalle oheislaitteelle niin sanottuja kiertokyselyviestejä. Näiden avulla päälaite voi varmistaa jokaisen laitteen olemassaolon dataväylässä. Mikäli oheislaitteella ei ole lähetettävää dataa, se vastaa päälaitteelle linjan vapautusviestillä (End Of Transmission, EOT).

Päälaite aloittaa kommunikaation oheislaiteille lähettämällä linjalle kyselyviestin (Enquery, ENQ):

PÄÄLAITE → OHEISLAITTEEN OSOITE,<ENQ>

Oikealla osoitteella varustettu oheislaite vastaa linjanvapautusviestillä:

OHEISLAITE → <EOT>

Tilanteessa, jossa oheislaite haluaa lähettää dataa päälaitteelle, tulee oheislaitteen odottaa ”puheenvuoroa”. Oheislaite saa puheenvuoron, kun se vastaanottaa omalle osoitteelleen suunnatun kiertokyselyviestin. Tämän kiertokyselyviestin lähettää päälaite. Näin ollen oheislaitteesta tulee väliaikainen päälaite ja päälaitteesta oheislaite.

Tässä tapauksessa alkuperäisellä oheislaitteella on mahdollisuus lähettää viestinsä. Viestin lähetys alkaa sanoman aloitusmerkillä (Start Of Header, SOH). Viestin sisällön kertominen aloitetaan viestin aloitusmerkillä (Start Of Text, STX). Viestin sisältöä pilkotaan erillisiin osiin. Viestin sisällön yksittäiset arvot aloitetaan käyttämällä aloitusmerkkiä (Unit Separator, US) ja arvojen erottelumerkkiä (Record Separator, RS). Viesti päätetään viestin lopetusmerkillä (End Of Transmission, ETX). Viestin lähetyksen jälkeen laitteiden ”roolit” palautuvat ennalleen. [1.]

Päälaite aloittaa kommunikaation lähettämällä linjalle:

PÄÄLAITE → OHEISLAITTEEN OSOITE,<ENQ>

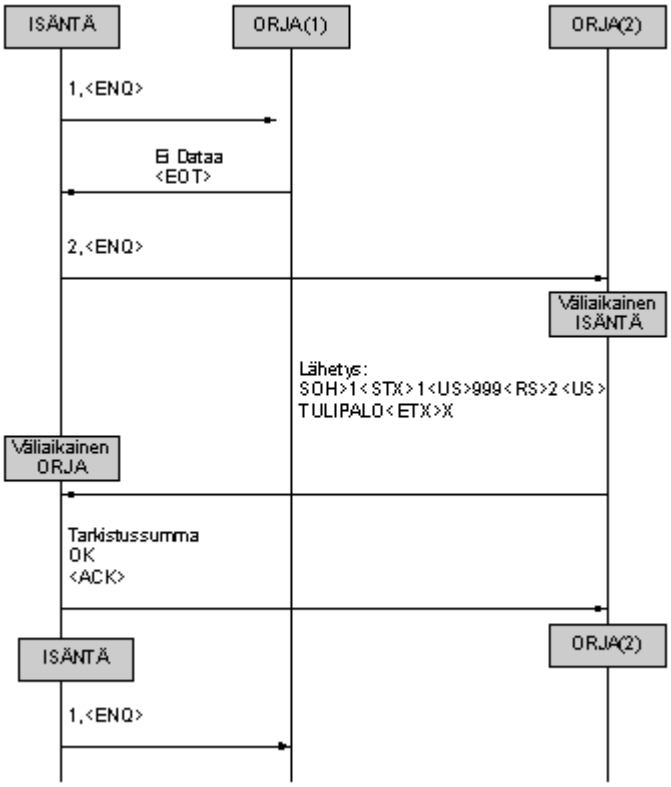
Koska oheislaitteella on dataa lähetettävänä, tapahtuu laitteiden välillä roolin vaihto:

OHEISLAITE muuttuu väliaikaiseksi PÄÄLAITE -laitteeksi
PÄÄLAITE muuttuu väliaikaiseksi OHEISLAITE -laitteeksi

Viestin lähetys:
Väliaikainen PÄÄLAITE → <SOH>1<STX>1 <US>999<RS>2 <US>TULIPALO <ETX>X

Viesti vastaanotettu:
Väliaikainen OHEISLAITE → <ACK>

Kun data on saatu vastaanotettua, vastaa alkuperäinen päälaite hyväksymisviestillä (Acknowledge, ACK) ja kommunikointi lopetetaan. Tässä vaiheessa laitteiden roolit vaihtuvat takaisin alkuperäiseen muotoonsa. Kuvassa 2 esitetään ESPA-kommunikaation kaaviokuva. [3.]



Kuva 2. ESPA-kommunikaatio. [3.]

3.2 Muut ominaisuudet

ESPA on ominaisuuksiensa puolesta monipuolinen protokolla. Nämä monipuoliset ominaisuudet saattavat selittää sen suosion esimerkiksi hoitajakutsu-, palo-, puhelin- sekä henkilöturvajärjestelmien piirissä. [1.]

ESPAn ominaisuuksiin kuuluu viestin virheentarkistus. Todellisen viestin lähetettyään ESPA-protokolla lisää vielä viestin loppuun niin kutsutun virheentarkistussumman. Tietoa vastaanottava laite voi laskemalla tarkistussumman itse ja vertailemalla sitä viestin lopussa löytyvään arvoon näin varmistua, että se on saanut viestin ehjänä itselleen. Mikäli tarkistussumma täsmää, lähetetään ACK-viesti. Mikäli tarkistussumma ei täsmää, on tiedonsiirrossa tapahtunut virhe. Kun tiedonsiirtoa halutaan yrittää uudestaan, suoritetaan se lähettämällä hylkäysviesti (Not Acknowledge, NAK). [3.]

4 Paloilmoitinjärjestelmät

Paloilmoitinkeskus

Paloilmoitinkeskukset liitetään aina isompaan hälytyskeskukseen. Tämä puolestaan hälyttää palokunnan paikalle. Paloilmaisimien lisäksi markkinoilta löytyy lisälaitteita, kuten sireenejä ja jälleenantomerkkivaloja. Kaikki lisälaitteet liitetään paloilmoitinjärjestelmän sydämeen eli paloilmoitinkeskukseen. Keskuksia on useita eri merkkejä. Tässä insinöörityössä käytettiin Ab Hedengren oy:n valmistamaa Prodex-merkkistä paloilmoitinkeskusta. Paloilmoitinkeskuksista hälytystieto saadaan välitettyä matkapuhelimiin, tietokoneisiin ja muihin sähköisiin viestintälaitteisiin. Tämä onnistuu käyttämällä niin kutsuttua Pronode-laitetta.

Pronode-laite

Tässä insinöörityössä käytettiin Prodex-valmistajan tarjoamaa Pronode-laitetta. Pronode-laite mahdollistaa palohälytystiedon siirron sarjaliikenteen kautta, ESPA 4.4.4:ää käyttäen. Palohälytyskeskus lähettää palohälytystiedon Pronode-laitteeseen, joka lähettää sen edelleen tietokoneen sarjaporttiin.

Pronode-laitteen avulla hälytysviestit voidaan lähettää myös tekstiviesteinä. Tekstiviestien lähetys vaatii erillisen GSM-modeemin käytön. GSM-modeemiksi Pronode-laitteeseen käy esimerkiksi Nokia 30 -modeemi. [5.]

Paloilmaisimien testaus

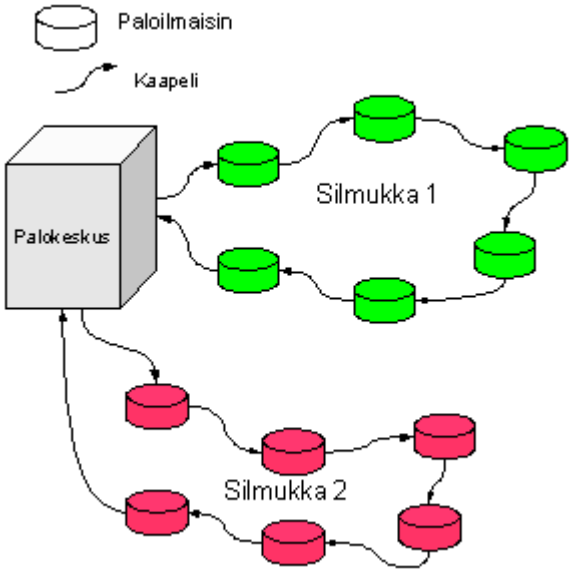
Paloilmoitinlaitteet, kuten muutkin nykypäivän elektroniikkalaitteet, ovat vioille alttiita. Siksi paloilmoitinjärjestelmiä suositellaan testattavan useasti vuoden aikana, jotta voidaan varmistaa henkilöturvallisuus tulipalon sattuessa.

Kun paloilmoitinjärjestelmä on fyysisesti asennettu, on järjestelmän käyttöönoton testauksen vuoro. Testauksessa paloilmaisimia ei testata aiheuttamalla rakennukseen tulipalo. Testauksessa paloilmaisin laukaistaan käyttämällä niin kutsuttua

testausmagneettia tai tulipalossa vapautuvan kaasun jäljitelmää. Näin voidaan varmistaa, että jokainen järjestelmään liitetty laite on kunnossa. Jokainen piste testataan yksi paloilmaisin kerrallaan. Testauksen jälkeen kunkin paloilmoitinjärjestelmän valmistajan omat asentajat tulevat suorittamaan paloilmoitinkeskuksen ohjelmoinnin. Tämä on tarkkaa työtä eikä virheitä sallita, kun kyseessä on ihmishenkien turvallisuus

Paloilmoitinkeskuksen ohjelmointi

Palokeskuksen ohjelmointi voidaan jakaa useampaan eri osa-alueeseen: kohde-, ryhmä- ja pistetietoihin. Kohdetiedossa kerrotaan rakennuksen osoite tai nimi, jotta tiedetään, missä rakennus fyysisesti sijaitsee. Ryhmätietoihin kirjataan rakennusosia tai yksittäisiä alueita. Tämä tieto on rakennuskohtainen, sillä jokainen eri rakennus on omanlaisensa yksikkö. Viimeiseksi ohjelmoidaan pistetieto, josta arkkitehtikuvaa käyttämällä nähdään huonenumerot sekä huoneiden tarkoitus, esimerkiksi siivouskomero, vahtimestarin tilat tai kanslia. Kuvassa 3 näkyy, kuinka paloilmaisinpisteet liitetään toisiinsa silmukan muotoisesti, jolloin ilmaisimet muodostavat katkeamattoman ketjun.

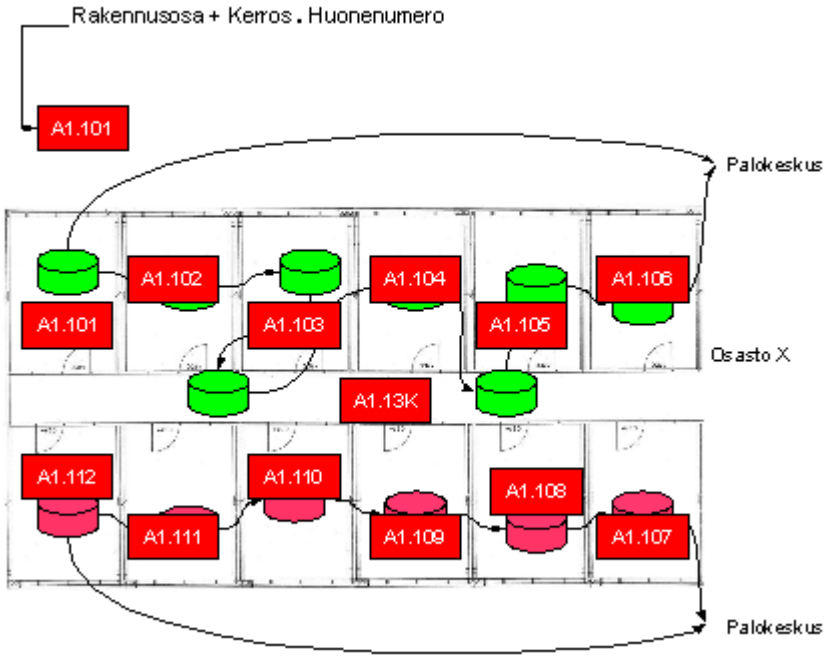


Kuva 3. Paloilmaisinpisteiden muodostamat silmukat.

Mikäli paloilmaisinpisteiden muodostama ketju katkeaa, syntyy virhehälytys ja laitevalmistajan asentajat kutsutaan paikalle korjaamaan vikaa. Yksittäiseen palosilmukkaketjuun voidaan liittää maksimissaan kuusitoista paloilmaisinta silmukkaa kohden. Jotta saadaan tietyn rakennusalueen ilmaisimet yhden ryhmätiedon alle, voidaan useampi silmukka liittää yhteen paloilmaisinryhmään. Näin esimerkiksi sairaaloissa eri osastot saadaan liitettyä omiin paloilmaisinryhmiinsä.

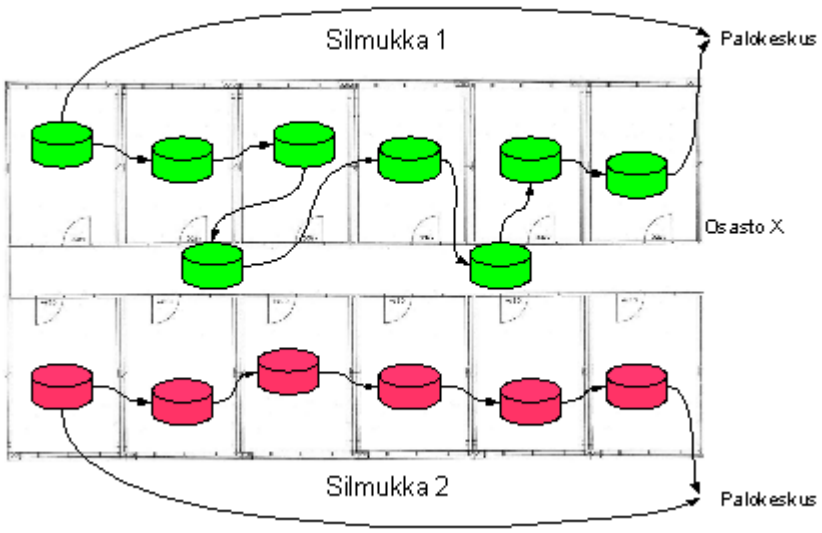
Hälytyskartat

Vasta kun kaikki paloilmaisimet on ohjelmoitu paloilmoitinkeskukseen, tehdään vielä jokaisen rakennuksen alueesta piste-, silmukka- ja ryhmätason hälytyskartat. Pistepohjaisiin hälytyskarttoihin sijoitetaan jokaisen yksittäisen pisteen huonenumero selkeästi. Kuva 4 esittää palohälytysjärjestelmän pistepohjaista hälytyskarttaa.



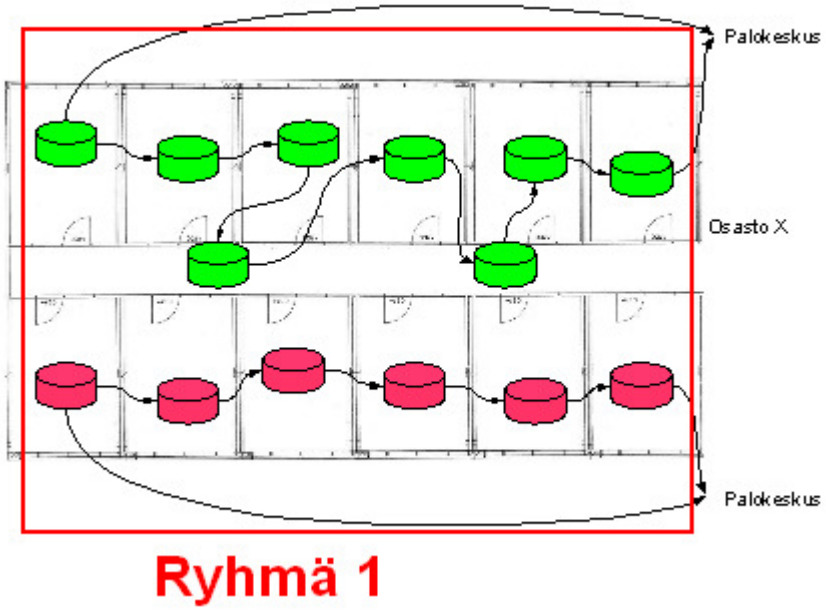
Kuva 4. Mallikuva pistepohjaisesta hälytyskartasta.

Rakennuksen alueista luodaan silmukkakartat, jotka kertovat mitkä paloilmaisinpisteet ovat saman silmukan sisällä. Kuva 5 esittää palohälytysjärjestelmän silmukkapohjaista hälytyskarttaa.



Kuva 5. Mallikuva silmukkapohjaisesta hälytyskartasta.

Ryhmäkuvissa nähdään, mitkä silmukat kuuluvat mihinkin ryhmiin. Kuva 6 esittää ryhmäpohjaista hälytyskarttaa.



Kuva 6. Mallikuva ryhmäpohjaisesta hälytyskartasta.

Virhehälytykset

Lähes kaikki palohälytykset ovat virhehälytyksiä. Näitä virhehälytyksiä syntyy monista syistä. Yleisimmät syyt virhehälytyksille löytyvät esimerkiksi ruuanlaitosta,

remonttityöstä sekä kosteudesta. Virhehälytystilanteissa palokunta saapuu paikalle ja palopäällikkö tarkistaa hälyttävän paloilmaisimen ja sen kunnon. Vasta kun ilmaisimen kunto on tarkistettu ja hälytyksen aiheuttaja selvitetty, hälytys voidaan kuitata. Nämä virheelliset hälytykset saattavat maksaa aiheuttajalleen noin tuhat euroa kerta.

5 EIRIS-ohjelmisto

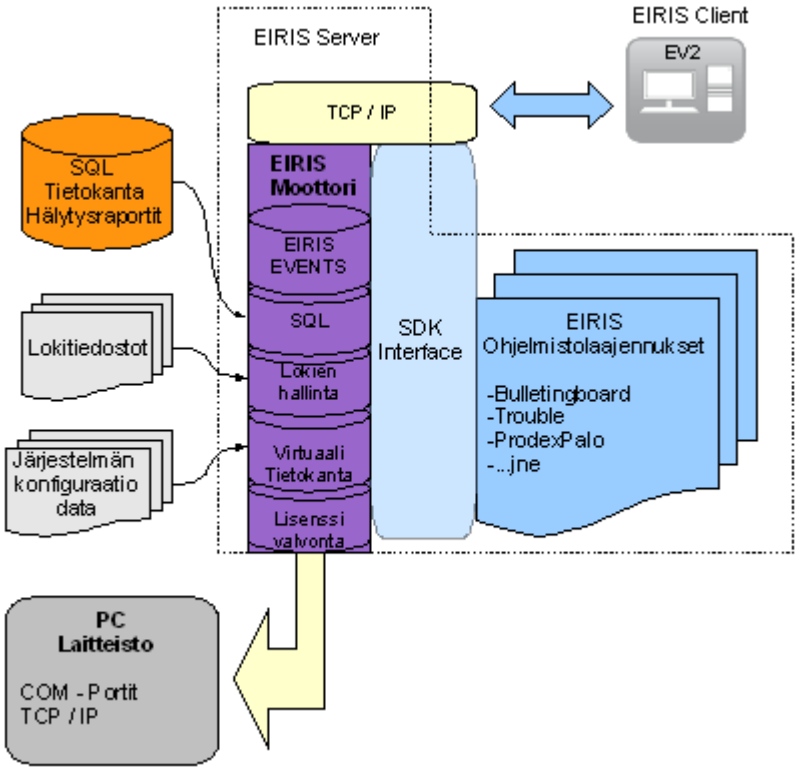
EIRIS-ohjelmisto on Visonic Technologies ltd:n kehittämä Windows-pohjainen turvajärjestelmien valvontaohjelmisto. EIRIS on suunniteltu pääsääntöisesti sairaaloihin, vankiloihin ja isoihin kiinteistöihin, joissa turvajärjestelmät ovat erittäin tärkeässä roolissa. Ohjelmisto on kehitetty valmiiksi yhdistämään Visonic Technologies ltd:n omat järjestelmät, kuten hoitajakutsu-, päällekarkaus-, videovalvonta-, kulunvalvonta- sekä irtaimistojenjäljitysjärjestelmä EIRISin tarkoituksena on saada kaikki yrityksen turvajärjestelmät yhden selkeän käyttöliittymän alle.

EIRIS-ohjelmisto muodostuu kahdesta osasta: EIRIS-server palvelinohjelmistosta ja EIRIS Viewer 2 -nimisestä hälytysten monitorointi-client-ohjelmasta, jolla ohjelmiston toimintaa myös konfiguroidaan. [4.]

5.1 Palvelinohjelmisto EIRIS-server

EIRIS-server -palvelinohjelmistoon sisältyy kaikki oleellinen konfiguraatiodata, kuten ohjelmistolaajennukset sekä lokien kirjoittaminen. EIRIS-ohjelmistossa on sisäänrakennettu SQL-tietokanta, jota käytetään hälytysraportointiin.

EIRIS-server muodostuu EIRIS-moottorista, joka on järjestelmän aivot. EIRIS-moottori ohjaa EIRIS-serveriin kuuluvia toimintoja, kuten hälytysraporttien SQL-tietokantaa, EIRIS-lisenssivalvontaa sekä liitettyjen järjestelmien konfiguraatiodataa. Palvelinohjelmistolla ei yksinään tee mitään, koska sen ylläpitäminen olisi teoriassa mahdotonta. Kuva 7 esittää EIRIS-ohjelmiston arkkitehtuurin yleiskuvaa.



Kuva 7. Yleiskuva EIRIS-ohjelmiston arkkitehtuurista

Lokien kirjoitus

Järjestelmän kaikki tapahtumat kirjoitetaan yhteen lokitiedostoon. Lokitiedostoihin kirjataan kronologisessa aikajärjestyksessä kaikki EIRIS-ohjelmiston sisällä tapahtuva tieto. Ohjelmistolaajennukset lähettävät käskyjä EIRIS-moottoriin ja käskevät sitä kirjaamaan haluttuja lokitietoja.

Hälytysraporttien SQL-tietokanta

Hälytysraporttien tietokanta on toteutettu erillisellä SQL-tietokannalla, johon kirjataan hälytystapahtumia. Hälytystapahtumista saadaan tärkeätä informaatiota asiakkaalle kytketyistä järjestelmistä.

Visonic-tuotteiden lisenssienvalvonta

Mikäli järjestelmän koko kasvaa hyvin suureksi, Visonic Technologies ltd:n valmistamien laitteiden ja ohjelmistolaajennuksien lisenssit maksavat käyttäjälleen

”pitkän pennin”. Lisenssimaksut ovat kertaluonteisia ja ne ovat sidoksissa EIRIS-ohjelmistoon liitettyjen laitteiden määrään.

EIRIS-ohjelmistolaajennukset

Jokainen EIRIS-ohjelmistoon liitetty turvajärjestelmä vaatii oman EIRIS-ohjelmistolaajennuksensa. Näitä EIRIS-ohjelmistoon liitettyjä turvajärjestelmiä voidaan hallita yksilöinä tai ne voidaan konfiguroida toimimaan keskenään. EIRIS-ohjelmistoon voidaan yhdistää liitettyjä järjestelmiä toimimaan keskenään ohjelmistotasolla.

5.2 Eiris Viewer 2 client -ohjelma (EV2)

Ylläpitoa ja hälytystenvalvontaa varten EIRIS:ssä on mahdollista käyttää Visonicin Technologies ltd:n luomaa EIRIS Viewer 2 client -ohjelma. EV2 on monipuolinen client-ohjelma. Sen avulla voidaan muokata olemassa olevien laitteiden toimintalogiikkaa ja EV2:n kykyä vastaanottaa hälytystietoja eri järjestelmistä.

EV2 on luotu hallinnoimaan ja vastaanottamaan hälytyksiä Visonic Technologies ltd EIRIS-ohjelmistolaajennuksista.

EV2:n avulla asentajat voivat konfiguroida EIRIS-ohjelmistoon logiikkaa ja hälytysgrafiikkaa laitekohtaisesti. EV2 pystyy myös vastaanottamaan EIRIS-ohjelmistoon hälytyksiä ja välittämään ne vain tietylle EV2 client -ohjelmille.

EV2:ssa on kolmitasoinen hälytysgrafiikka. Kolmena eri tasona voidaan käyttää esimerkiksi alueen yleis-, yksittäisrakennus- sekä arkkitehtikuvaa. Kaikki ohjelmoidut hälytyspisteet voidaan liittää hälytysgrafiikan eri tasoihin.

5.3 EIRIS–ohjelmistolaajennusten kehitystyökalu

Kaikkien järjestelmien ohjelmistolaajennukset kirjoitetaan käyttäen EIRIS SDK -työkalua (Software Development Kit, SDK). EIRIS SDK -työkalu mahdollistaa ohjelmistolaajennukset ja client-pohjaisten ohjelmistolaajennusten ohjelmoinnin EIRIS-ohjelmistoon, C++-ohjelmointikieltä käyttäen.

SDK-työkalu toimii useiden eri ohjelmointikielien kanssa. Näistä C++-ohjelmointikieli on monipuolisin. Sen avulla saadaan EIRIS-ohjelmistoon kaikki toiminnallisuudet esiin. Ohjelmointialustana täytyi käyttää Microsoft Visual C++ 6.0 -ohjelmistoa.

5.4 EIRIS-objektit

Ohjelmistolaajennuksissa on erilaisia objekteja, yleensä keskus-, hallinta- ja laiteobjekteja. Keskusobjekti liittää fyysisen järjestelmän EIRIS-ohjelmistoon. Hallintaobjektit ohjaavat EIRIS-tapahtumia ja muita objekteja EIRIS-ohjelmistossa. Laiteobjektit ovat liitettyjen järjestelmien yksittäisiä laitteita tai laitekokonaisuuksia.

EIRIS-ohjelmisto sisältää useita valmiita objekteja, joita tässä työssä hyödynnettiin, joten ”pyörää ei tarvinnut keksiä uudelleen”. Tämän insinöörityön ohjelmistolaajennuksen toiminnan kannalta tärkeimmät valmiiksi rakennetut objektit olivat Sarjaportti-, Bulletingboard- ja Trouble-objektit.

Sarjaportti-objekti määrittelee PC:n sarjaportin asetukset. Sarjaportti-objekti mahdollistaa esimerkiksi ESPA-datan lukemisen Pronode-laittelta EIRIS-ohjelmistoon.

Bulletingboard-objekti lähettää hälytystietoa oikeassa formaatissa Telepulssi Oy:n käytävänäyttöihin. Bulletingboard-objektin toimintaa hyödyntämällä voidaan näyttötaulut myös kuitata hälytystilasta.

Trouble-objekti oli hyvin tärkeä osa insinöörityötäni. Trouble-objekti oli suunniteltu aktivoimaan muita objekteja. Tämän objektin avulla palohälytykset näkyvät EV2:ssa ja käytävänäytöissä.

6 Palohälytys-ohjelmistolaajennuksen suunnittelu ja toteutus

6.1 Projektin alkuvaiheet

Insinöörityön alustava määrittely aloitettiin projektin varhaisessa vaiheessa. Jo projektin alkuvaiheessa tiedettiin, että tullaan käyttämään Telepulssi Oy:n käytävänäyttöjä, joihin palohälytystietoja välitetään. Projektin tärkeimmäksi haasteeksi osoittautui kysymys siitä, kuinka toteutetaan hälytysten välitys palohälytyskeskukselta käytävänäyttöihin.

Projektin alussa tutkittiin Meilahden yhteispäivystyssairaalan työmaa-asiakirjoista minkä valmistajan palohälytyskeskus kohteeseen oli suunniteltu. Kohteeseen oli valittu Ab Hedengren oy:n Prodex-merkkinen palokeskus.

Prodex-palokeskuksen valmistajalta tiedusteltiin mitä ratkaisuja yrityksellä on tarjota palohälytystietojen välitykseen palokeskukselta. Yrityksestä kerrottiin Pronode-laitteesta, joka mahdollistaisi hälytysten siirron ESPA-protokollan avulla tietokoneeseen. Ohjelmistolaajennuksen kehityksen avuksi lainattiin Ab Hedengren oy:n Pronode-laitetta. Laitteen mukana tuli hyödyllinen palohälytyssimulaattoriohjelmisto, jonka tarkoituksena on simuloida Prodex-palohälytyskeskusta. Pronode-laite mahdollistaa palohälytystiedon siirron ESPA 4.4.4 -protokollan kautta.

Insinöörityössä päädyttiin käyttämään PC-pohjaista vaihtoehtoa Pronoden-laitteen kanssa. PC:n korvaajaksi Telepulssi Oy:llä olisi ollut useita erilaisia vaihtoehtoja, koska yrityksen valmistamat elektroniikkalaitteet tukevat myös ESPA-protokollaa. Näiden laitteiden muokkaaminen projektin tarkoitusta varten olisi ollut liian työlästä, tämän takia päädyttiin käyttämään PC-pohjaista ratkaisua. PC- ja sarjaporttiliitäntä oli Telepulssi Oy:lle jo entuudestaan tuttua. Tuntui luonnolliselta käyttää tässäkin projektissa tutuksi tullutta sarjaporttiratkaisua. Haasteeksi muodostui, rakentaako oma ohjelma vai käyttääkö valmista ohjelmistoratkaisua palohälytysten vastaanottamiseen. Pitkän mietinnän ja harkinnan jälkeen päädyttiin Visonic Technologies ltd:n valmistamaan EIRIS-ohjelmistoon.

EIRIS-ohjelmisto oli tullut Telepulssille hyvinkin tutuksi Visonic Spider -henkilöturvahälytysjärjestelmien myötä. Visonic Technologies ltd tarjosi meille SDK-työkalua, jonka avulla voisimme liittää palohälytyskeskuksen hälytykset EIRIS-ohjelmistoon.

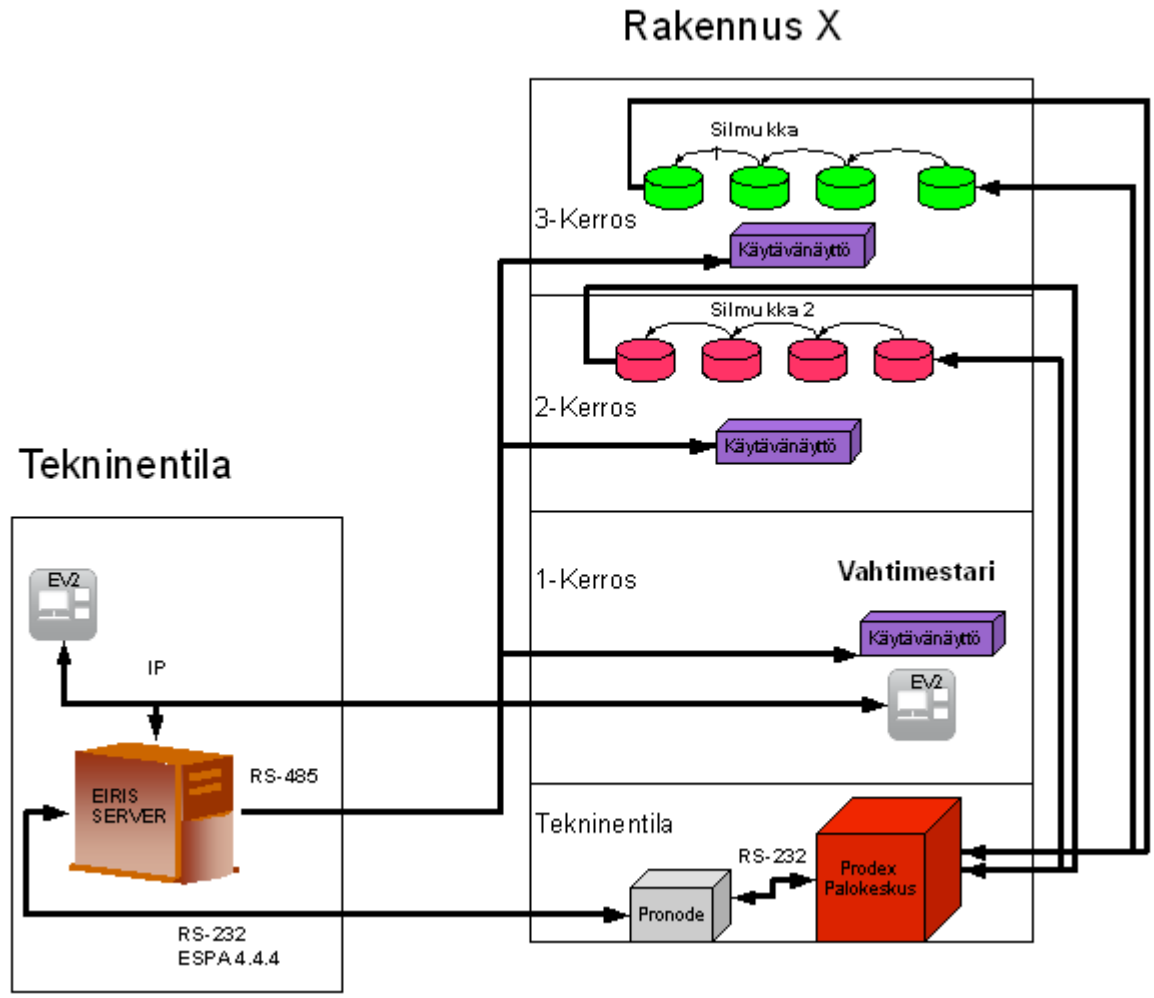
Ennen SDK-työkalun käyttöönottoa oli perehdyttävä EIRIS-ohjelmoinnin maailmaan. Visonic Technologies ltd:n tarjoamaa koulutusta varten hankittiin Microsoft Visual C++ 6.0 -ohjelma. Koulutuksessa oli apuna Telepulssi Oy:n hoitajakutsujärjestelmän keskusyksikkö sekä niin sanottuja potilaskojeita. Näin tiedonsiirtoa EIRIS-ohjelmiston ja laitteiden välillä voitiin testata.

Nyt Telepulssi Oy:lla oli perustiedot, kuinka luodaan EV2-tyyppisiä, client-pohjaisia ohjelmistolaajennuksia EIRIS-ohjelmistoon. Yksi syy valita juuri EV2 oli aika. Täysin uusi client-ohjelmistonlaajennus olisi vienyt aivan liikaa aikaa. Tärkeimpänä asiana opittiin luomaan EIRIS-ohjelmistoon toimiva ohjelmistolaajennusluuranko.

Tässä projektissa oli paljon asioita, jotka eivät olleet Telepulssi Oy:lle entuudestaan tuttuja, kuten C++- ja olio-ohjelmointi sekä EIRIS-ohjelmistolaajennusten ohjelmointi.

6.2 Järjestelmän yleiskuvan määrittely

Ennen kuin päästiin suunnittelemaan ohjelmistolaajennusta, oli aluksi määriteltävä järjestelmän yleiskuva. Sen tekeminen oli projektin helpoimpia kohtia. Lähestymistä mietittiin seuraavanlaisesta näkökulmasta. Hälytykset piti saada näkyviin rakennuksessa ja kuitattua useissa kerroksissa ja rakennusosissa. Hälytys piti myös saada näkyviin EV2:ssa. Kuva 8 esittää saatua mielikuvaa järjestelmän yleiskuvasta.



Kuva 8. Järjestelmän yleiskuva.

6.3 Ohjelmointimallin valinta

Varsinaisen ohjelmointikehitystyön alkaessa ei tiedetty EIRIS-ohjelmiston rajoitteista ja potentiaalista. C++-ohjelmointikielenä oli Telepulssi Oy:lle uusi. Joten täytyi valita ohjelmointimalli, joka helpottaisi nopeaa järjestelmän analysointia ja uuden ohjelmointikielen opiskelua. Loogiselta vaihtoehdolta tuntui valita prototyyppi-ohjelmointimalli.

Ohjelmointimalli auttaa tekemään nopeita määrittelyjä, suunnitelmia ohjelmiston toimintaan ja näkemään työn tulokset prototyyppinä. Prototyyppi-ohjelmointimalli antoi mahdollisuuden oppia EIRIS-ohjelmiston toimintaa ja C++-ohjelmointikieltä, tosin yrityksen ja erehdyksen kautta.

6.4 Prototyyppien ohjelmointi

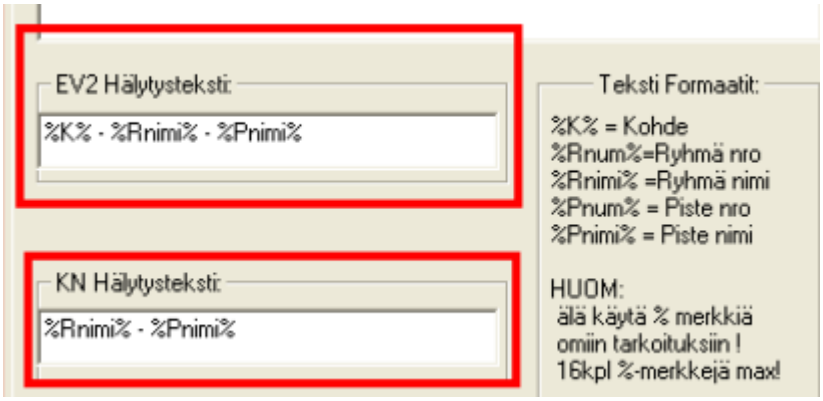
Varsinaisen ohjelmoinnin alkaessa kaikki laitteet ja ohjelmistot oli hankittuina: PC varustettuna Microsoft Visual C++-kehitysympäristöllä, Pronode-laite, Prodex-palohälytyskeskusta simuloiva ohjelma sekä käytävänäyttö.

ESPA-kommunikaatio

Prototyyppien suunnittelussa otettiin niin sanottuja välietappeja. Ensimmäisenä etappina oli saada Pronoden lähettämät ESPA-viestit sisään EIRIS-ohjelmistoon. Tätä varten sarjaliikenne EIRISin sisällä ohjattiin Palohälytys-ohjelmistolaajennukseen. Tämä mahdollistettiin käyttämällä valmista Sarjaliikenne-objektia nimeltään Serial Interface. Sarjaliikenneobjekti mahdollistaa ESPA-viestien vastaanottamisen tietokoneen sarjaportin kautta. Useita päiviä käytettiin ja monia erilaisia kommunikointiprototyyppejä luotiin, ennen kuin ESPA-kommunikointi toimi protokollan mukaisella tavalla. Tämän jälkeen kirjoitettiin loki -toiminnallisuus saapuvien ja lähtevien ESPA-viestien sisällöstä. Näin lokeista voidaan tarkistaa mitä viestejä järjestelmä on todellisuudessa vastaanottanut.

Hälytys- ja kuittausominaisuus

Seuraava merkittävä välietappi oli luoda Palohälytys-ohjelmistolaajennukseen toiminnallisuus, jonka avulla palohälytysviestit lähetettäisiin käytävänäyttöihin. Jälleen yrityksen ja erehdyksen kautta löydettiin lopulta keino käyttää Trouble-objektia aktivoimaan Bulletingboard-objekti. Bulletingboard-objekti lähettää aktivoituessaan palohälytystietoa käytävänäyttöihin. Positiivisena yllätyksenä huomattiin, että käyttämällä Trouble-objektia saatiin myös palohälytysviestit välitettyä EV2:een. Tässä vaiheessa syntyi idea tarjota käyttäjille mahdollisuus valita, mitkä tiedot halutaan välittää käytävänäyttöihin ja mitä EV2:een. Kuva 9 esittää, kuinka käyttäjällä on vapaus valita hälytystekstien formaatti.



Kuva 9. Tekstiformaatin valinta käytävänäyttöihin ja EV2:een.

Hälytysten kuittausominaisuuden rakentaminen oli tosi asiassa vaikeampaa kuin itse palohälytyksen aktivointi EIRIS-ohjelmistossa. Tämä rakentamisen vaikeus johtui Trouble-objektin toimintalogiikasta. EV2 mahdollistaa hälytysten kuittauksen hälytysnäytöltä.

Touble-objektin laaja tutkimusprosessi auttoi ymmärtämään paremmin EIRIS-ohjelmiston peruskomponenttien toimintalogiikkaa. Yhden objektin perusteellinen tutkinta vei ajallisesti viikkoja. Tämä tutkimusprosessi auttoi hälytys- ja kuittauslogiikan toteuttamisessa. Uuden kuittauslogiikan myötä hälytysten kuittaaminen oli mahdollista EV2:sta, ulkopuolisista kytkimistä sekä ohjelmasta itsestään.

Tämä hälytys- ja kuittauslogiikka oli tärkeä läpimurto projektissa. Suunnittelutyön aikana mietittiin muun muassa, kuinka haluttiin tulkita palohälytyksiä Pronodesta. Järkevimpinä vaihtoehtoina mietittiin piste- tai ryhmäkohtaisia hälytyksiä. Ryhmäkohtaisiin hälytyksiin päädyttiin syistä, jotka perustelen myöhemmin. Päätös ryhmäkohtaisista palohälytyksistä muutti ohjelman toimintaa niin paljon, että ohjelmistolaajennus jouduttiin luomaan kokonaan uusiksi käyttäen olemassa olevia toteutettuja toiminnallisuuksia vanhoista prototyypeistä.

Palokeskus- ja Ryhmienhallinta-objekti

Uuteen ohjelmistolaajennukseen luotiin kaksi erillistä objektia, jotka nimettiin Palokeskus- ja Ryhmienhallinta-objekteiksi.

Palokeskus-objekti rakennettiin kommunikoimaan Pronoden kanssa ja yhdistämään Ryhmienhallinta-objektit tiettyyn palokeskusobjektiin. Tämä Palokeskus-objekti määrittelee, missä formaatissa hälytykset näkyvät EV2:ssa ja käytävänäytöissä.

Ryhmienhallinta- ja Trouble-objektien avulla pystyttäisiin hallitsemaan mihin näyttötauluihin ja EV2:iin mitäkin palohälytystietoja lähettäisiin. Ryhmienhallinta-objektin syntymä antoi vapauden järjestelmän asentajille profiloida palohälytyksiä ja kuittauksia tarkasti asiakkaan toiveiden mukaisesti.

Yhteys Pronode-laiteeseen

Palokeskusobjektin määrittelyvaiheessa yksi tärkeimmistä ominaisuuksista oli täysin unohtunut – mitä jos yhteys Pronode-laitteeseen katkeaa? Pronode-laitteen toiminnallisuuksia ei voitu muuttaa, joten oli keksittävä vaihtoehtoinen tapa saada tieto yhteyden katkeamisesta EIRIS-ohjelmistoon ja Pronoden välillä. Tämän ongelman ratkaisu oli Pronoden ESPA-päälaiteominaisuus.

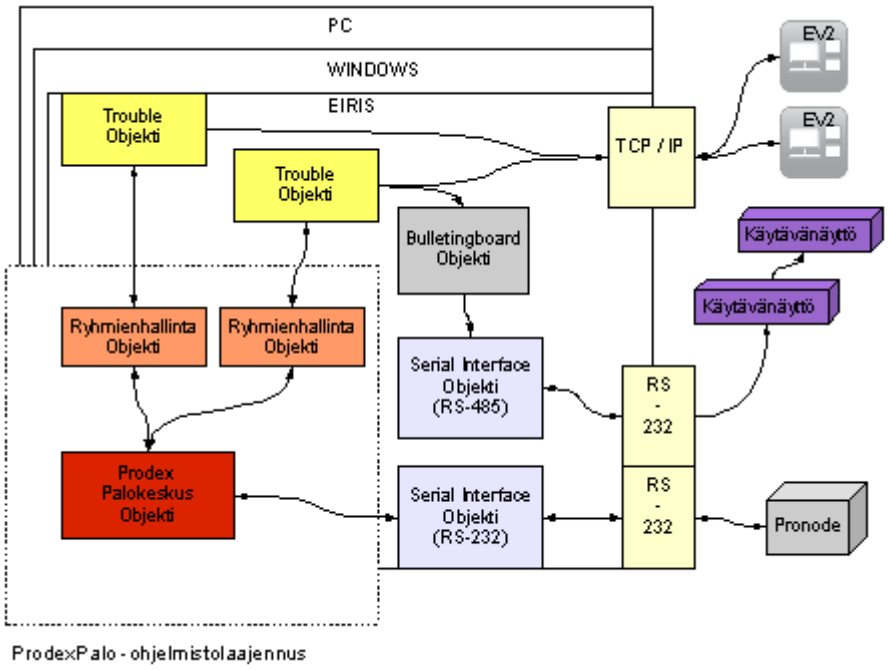
ESPA-päälaitteen toiminnallisuuteen kuuluu lähettää toistuvasti ENQ-viestejä. Näitä ENQ-viestejä monitoroimalla voitiin pitää kirjaa, onko päälaite vielä kytkettynä EIRIS-ohjelmistoon. Mikäli Palokeskus-objekti ei saa valitulta Pronode-laitteelta kuudenkymmenen sekunnin aikana yhtään ENQ-viestiä, aktivoituu ”keskus ei vastaa” -hälytys. Tämän toiminnallisuuden lisääminen johti Palokeskus-objektin rakenteen täydelliseen uusimiseen. Tämän monitorointitoiminnallisuuden kehittäminen vei ajallisesti useita viikkoja.

Käytävänäyttöjen kehitys

Telepulssi Oy:n käytävänäyttöjen rajoitteena oli näyttää vain kahdeksan yhtäaikaista merkkiä kerrallaan. Tämä rajoitteen poistaminen palohälytysteksteistä oli merkittävä laajennus palohälytystietojen käytettävyyden kannalta. Näin kahdeksan merkin sijaan pystyttiin näyttämään useita kymmeniä merkkejä. Olemassa olevien käytävänäyttöjen toiminnallisuutta muokattiin vierittämään hälytystekstiä näytöllä uudestaan ja uudestaan.

Viimeinen prototyyppi

Kun tärkeimmät toiminnallisuudet oli kartoitettu ja luotu, täytyi ottaa uusi näkökulma – asentajien näkökulma. Tarkoituksena oli tehdä ohjelmistolaajennus dynaamiseksi ja helppokäyttöiseksi. Palohälytys-ohjelmistolaajennuksesta poistettiin myös ”inhimillisen virheen” mahdollisuus. Eräänä esimerkkinä on keskusten numeroinnin automaattisuus Palokeskus-objektien lisäysvaiheessa. Kun objekti ensimmäisen kerran avataan, järjestelmä tarkistaa, kuinka monta vanhaa Palokeskus-objektia on jo lisätty, ja antaa uudelle lisätylle objektille seuraavan vapaan numeron. Kuvassa 10 näkyy, kuinka Palohälytys-ohjelmistolaajennus sijoittuu EIRIS-ohjelmistoon.



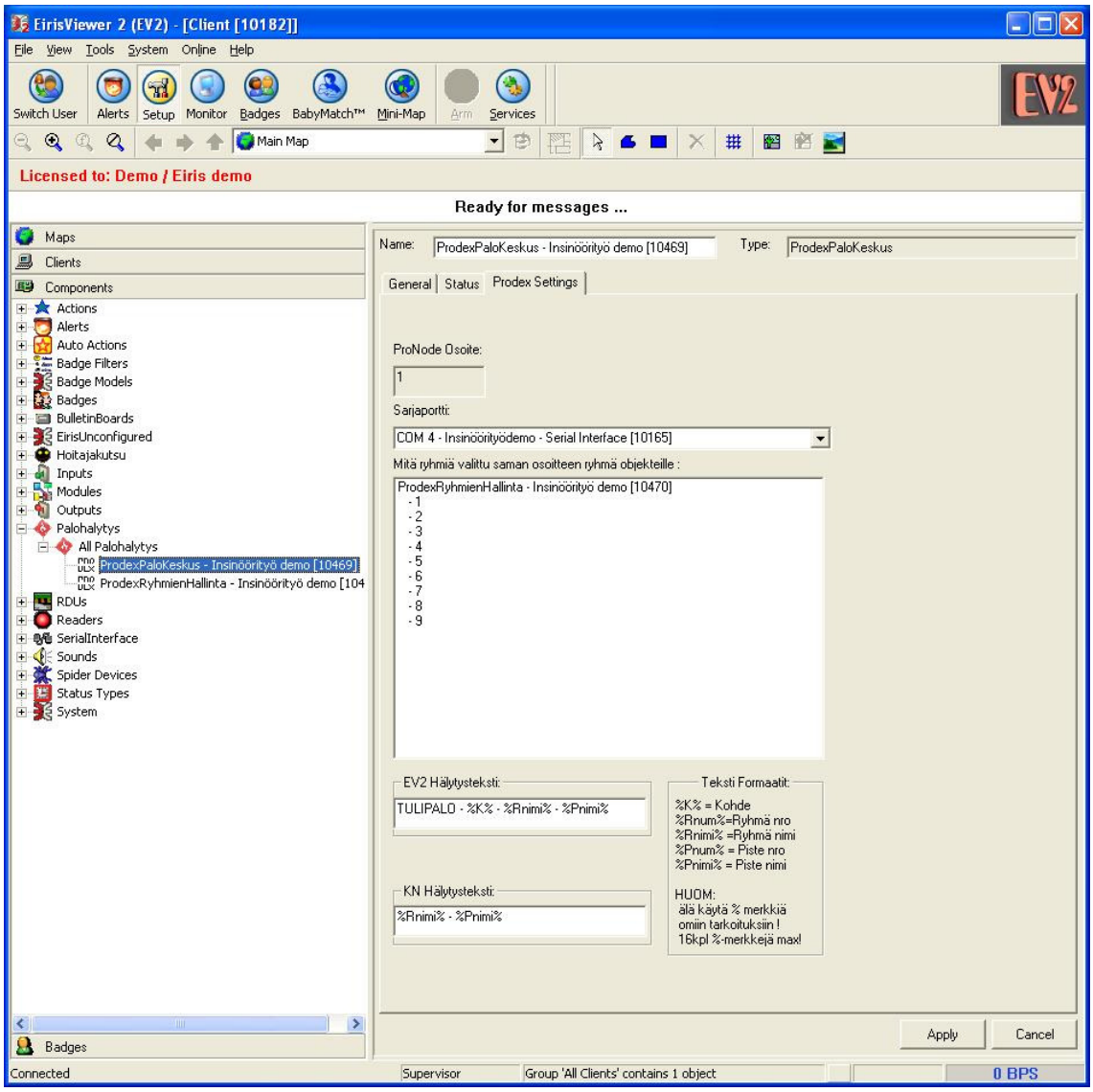
Kuva 10. EIRIS-objektien välinen korrelaatio

6.5 Lopullinen tuote

Palohälytys-ohjelmistolaajennuksen ensimmäisessä julkisessa versiossa syntyi kaksi erillistä objektia: Palokeskus- ja Ryhmienhallintaobjektit. Koko ohjelmistolaajennus sisältää kahdesta kolmeentuhanteen riviä itse ohjelmoitua koodia.

Palokeskus-objekti

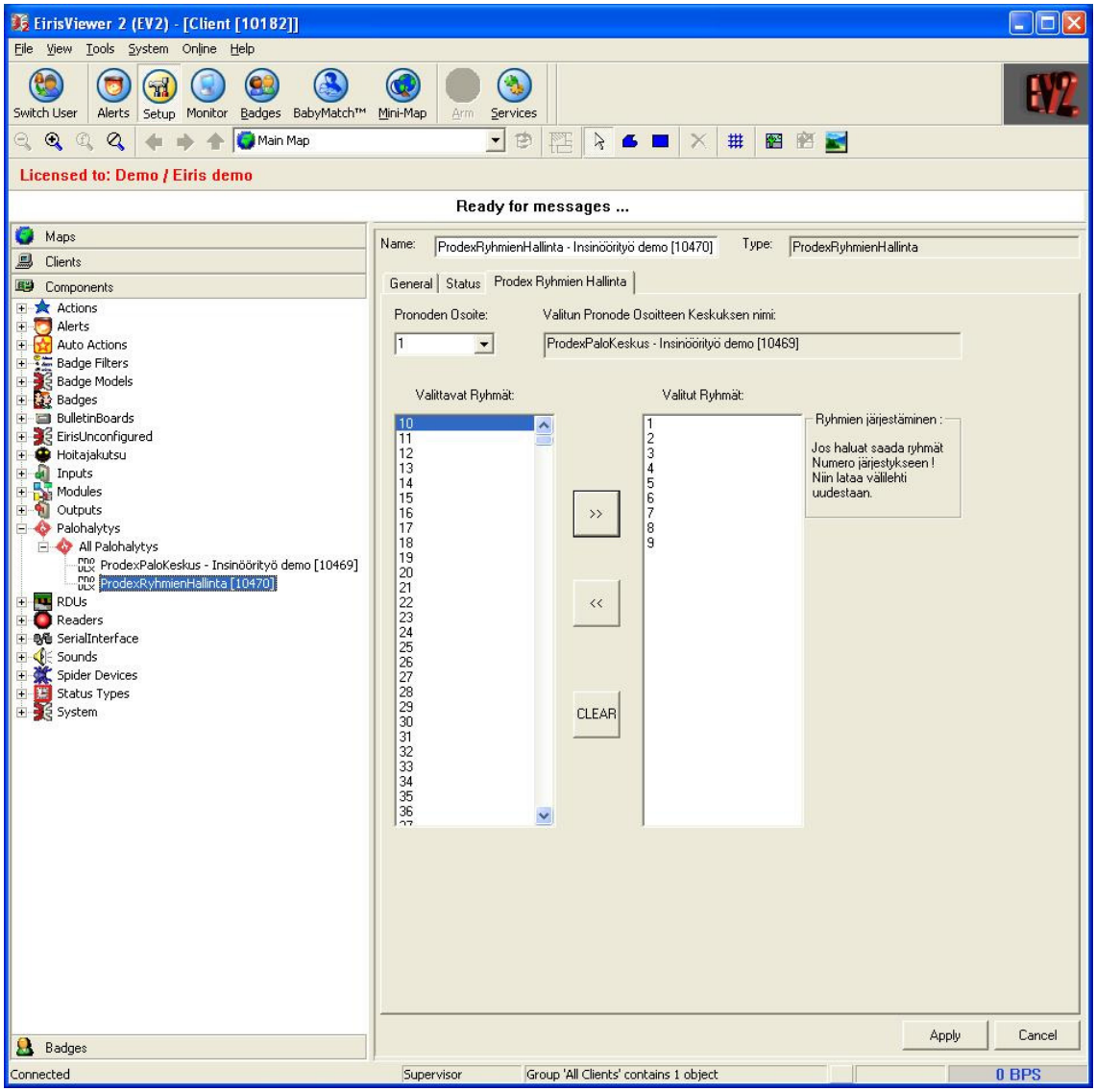
Palokeskus-objektin ominaisuuksiin kuuluu muun muassa keskusnumeron automaattinen valinta, sarjaportin valinta ja hälytystekstien formaattien muokkaaminen. Se myös tulostaa kaikki palokeskusobjektiin liitetyt Ryhmienhallinta-objektit sisältöineen näytölle ja näin helpottaa myös asentajien työtä. Kuvassa 11 näytetään, miltä Palokeskus-objekti näyttää EV2:ssa.



Kuva 11. Palokeskus-objektin käyttöliittymä EV2:ssa.

Ryhmienhallinta-objekti

Ryhmienhallinta-objektin ominaisuuksiin kuuluu Palokeskus-objektin osoitteen valinta ja ryhmänumeroiden valinta. Kuvassa 12 esitellään miltä Ryhmienhallinta-objekti näyttää EV2:ssa.



Kuva 12. Ryhmienhallinta-objektin käyttöliittymä EV2:ssa.

7 Palohälytys-ohjelmistolaajennuksen testaus

Prototyyppi-ohjelmointimallin mukaisesti ohjelmistolaajennusta testattiin jokaisen pienimmänkin muutoksen jälkeen.

Testiympäristö

Testijärjestelmän kokoonpanona toimi PC, jossa käytettyinä ohjelmina toimivat EIRIS-server, EV2, Prodex-palohälytyssimulaattori, useita sarjaportin analysointityökaluja sekä erillinen muistivuotoja tarkkaileva ohjelma. Tietokoneeseen kytkettyjä laitteita olivat Pronode-laite, kaksi käytävänäyttöä, hoitajakutsujärjestelmä ja Visonicin Spider-henkilöturvajärjestelmä.

Testiympäristössä haluttiin myös saada selvyys Palohälytys-ohjelmistolaajennuksen yhteensopivuudesta muiden ohjelmistolaajennusten kanssa.

EIRIS-ohjelmiston testauksen ongelmat

EIRIS-serverin testaus yhtä aikaa ohjelmiston kehittämisen kanssa on hyvin haastavaa. Tämä johtuu EIRIS-serverin epävakaudesta ajettaessa sitä Microsoft Visual C++-ohjelmointikehitysalustalla. Tämä epäkohta hidasti testausprosessia huomattavasti. Testaaminen piti suorittaa luomalla ohjelmistolaajennuksesta toimiva dll-tiedosto (Dynamic Link Libary, DLL) ja käynnistää EIRIS-server ilman Visual C++-kehitysalustaa. Tätä toimenpidettä tehtiin parhaillaan useita kertoja päivässä. Suurin osa työpäivästä kului lähinnä ohjelmistolaajennuksen testaamiseen.

Tulikoe

Viimeisenä tulikokeena Palohälytys-ohjelmistolaajennus jätettiin pyörimään yötä päivää kuukauden ajaksi. Kaikkia EIRIS-ohjelmistoon liitettyjä laitteita testattiin joka päivä. Kytketyistä järjestelmistä aiheutettiin hälytys- ja kuittaustapahtumia. Ohjelmistolaajennuksen muistivuotoja tarkkailtiin erillisellä muistinvalvontatyökalulla. Testien aikana yhtään muistivuotoa ei ohjelmistossa havaittu.

Testit tuottivat positiivisia tuloksia. Järjestelmään aiheutetut hälytykset toimivat käytävänäytöissä ja EV2:ssa. Jokainen EIRIS-ohjelmistoon liitetty hälytysjärjestelmä toimi rinnakkain. Tärkein tulos oli, että Palohälytys-ohjelmistolaajennus toimi koko testauksen ajan virheettömästi. Tämä tulikoe antoi varmuuden ohjelmistolaajennuksen toimivuudesta ennen virallista kenttäkokeilua.

8 Lopputulokset ja päätelmät

Insinöörityöni oli todella haasteellinen ja monipuolinen. Työn alkaessa perustietoni C++- ja olio-ohjelmoinnista rajoittuivat vain yhteen ainoaan koulussa käytyyn kurssiin. Sain opetella myös täysin uuden ohjelmointikielen sekä kartoittaa itselleni tuntemattoman järjestelmän vahvuuksia ja heikkouksia.

Työssä käytetty prototyyppiohjelmointimalli oli paras mahdollinen tapa opetella nopeasti uusi ohjelmointikieli. Näin vieraan järjestelmän toimintaa pääsee testaamaan heti prototyypin valmistuttua.

Prototyyppiohjelmointimallin käyttäjälle saattaa nousta mieleen seuraavia kysymyksiä: Milloin prototyyppien kehittäminen pitäisi lopettaa? Onko toiminnallisuus tarpeeksi hyvin toteutettu? Hylätäänkö prototyyppi vai jatkokehitetäänkö se tuotantokäyttöön? Ohjelmointityön tilaaja voi myös tulkita prototyypin jo valmiiksi järjestelmäksi, eikä ymmärrä miksi sitä pitäisi vielä hioa. [2.]

EIRIS on todella vakaa järjestelmä kovankin rasituksen alla. Järjestelmä on skaalattavuudeltaan oikein rakennettu, joten ilman ongelmia järjestelmään kytkettävien laitteiden lukumäärät voivat olla kymmenissä tuhansissa.

Järjestelmän käyttöliittymää ei kuitenkaan ole suunniteltu asentajien monipuoliseksi työkaluksi. Tämä ohjelmiston epäkohta aiheuttaa sen, että kymmenien järjestelmään liitettyjen turvajärjestelmien samanaikainen ylläpito muuttuu todella hankalaksi.

EIRIS-ohjelmistoon tehtiin Palohälytys-ohjelmistolaajennusta varten omat käyttöohjeet. Ohjeisiin kuului Pronode-laitteen ohjelmointiopas EIRIS-ohjelmistoyhteensopivaksi sekä Palohälytys-ohjelmistolaajennuksen integraatio-opas EIRIS-ohjelmisto. Käyttöohjeista tuli laajat, monipuoliset ja hyvin kuvitetut. Uskallan luvata, että näitä ohjeita noudattaen EIRIS-ohjelmistossa ei voi tehdä virheitä.

Tärkein virstanpylväs koko projektissa oli pistepohjaisten palohälytystietojen käsittelyn hylkääminen ja siirtyminen ryhmäkohtaiseen ajatteluun. Vaikka pistepohjainen ajattelu olisi mahdollistanut huonekohtaisen hälytysgrafiikan käytön EV2:ssa, tämä olisi aiheuttanut liikaa muita murheita. Palohälytysjärjestelmissä on satoja tai tuhansia

hälytyspisteitä ja tämä olisi tehnyt EIRIS-ohjelmiston käytettävyydestä ja ylläpidosta hankalan. Joka kerta, kun olemassa olevaan palokeskukseen lisättäisiin uusi paloilmaisin, pitäisi myös uusi paloilmaisinpiste lisätä EIRIS-ohjelmistoon. Tällöin jokaisesta pisteestä olisi luotava oma pistekohtainen objektinsa. Näin ryhmätasoinen ajattelu mahdollistaa paloilmaisimien lisäämisen ryhmiin ilman, että EIRIS-ohjelmiston päässä tarvitsee tehdä yhtään mitään.

Pidän Palohälytys-ohjelmistolaajennuksen lopputuloksesta, joka on asentajaystävällinen ja tarjoaa loppukäyttäjille helposti luettavan palohälytystiedon käytävänäyttöihin. Itse olisin toivonut voivani käyttää enemmän aikaa ohjelmiston puhtaaksikirjoittamiseen – mutta toimiihan ohjelmisto nykyisilläkin koodeilla.

Palohälytys-ohjelmistolaajennuksesta löytyy paljon parannettavaa. Kuitenkin kirjoitettu koodi on toimivaa, mutta ei helposti uudelleen käytettävää. Jos ohjelmiston määrittelyt olisi tehty olio-ohjelmointi mielessä pitäen, tulokseksi olisi saatu luokkakaaviot ohjelmiston eri osa-alueista ja niiden ominaisuuksista.

Palohälytys-ohjelmistolaajennus on siis asentajaystävällinen ja ominaisuuksiltaan monipuolinen. Se toimii yhteensopivasti Prodex-palokeskusten ja muiden EIRIS-ohjelmistoon asennettujen ohjelmistolaajennusten kanssa. Palohälytys-ohjelmistolaajennus tekee juuri sen mitä se on suunniteltu tekemään.

Lähteet

1 ESPA 4.4.4 2011. Verkkodokumentti. <http://www.bxo.se/en/technical-information-about-alarms/19-armprotokoll/129-espa-4-4-4.html> Luettu 25.2.2011.

2 Ohjelmistotuotanto 2011. Verkkodokumentti. <http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento2-bw.pdf> Luettu 20.3.2011.

3 ESPA 4.4.4; Serial Data Interface For Paging Equipment 2003. Verkkodokumentti. <http://www.gscott.co.uk/ESPA.4.4.4/> Luettu 10.7.2008.

4 EIRIS-Data-Sheet 2009. Verkkodokumentti. <http://www.visonictech.com/Open-Docs/Eiris-Software/EIRIS-Data-Sheet.pdf> Luettu 4.1.2008.

5 Nokia 30 UserGuide. Verkkodokumentti. <http://nds1.nokia.com/phones/files/guides/Nokia_30_UG_en_.pdf> Luettu 28.3.2011.

6 Telepulssi OY 2011. Verkkodokumentti. <http://www.cylex.fi/yritys/telepulssi-oy-10543625.html> Luettu 06.02.2011.