Ville Vuorela

KIINTEISTÖAUTOMAATIOJÄRJESTELMIEN ETÄKÄYTTÖLIITTYMÄSOVELLUS

Tietotekniikan koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KIINTEISTÖAUTOMAATIOJÄRJESTELMIEN
ETÄKÄYTTÖLIITTYMÄSOVELLUS

Vuorela, Ville
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Toukokuu 2015
Ohjaaja: Trast, Ismo
Sivumäärä: 34
Liitteitä: 0

Asiasanat: Kiinteistöautomaatio, Valvomo, Käyttöliittymä, C#

Tässä opinnäytetyössä käydään läpi keväällä 2015 luodun ohjelman kehitystä. Ohjelma tehtiin helpottamaan valvomotyöntekijän työtä kiinteistöautomaatiojärjestelmän keskusvalvomossa antamalla käyttäjälle keino käynnistää etäyhteys haluttuihin ennaltamäärättyihin kohteisiin karttapohjan päällä olevia nappeja painamalla. Tämän lisäksi käyttäjällä piti olla mahdollisuus lisätä, poistaa ja muokata ohjelmassa olevia kohteita. Vastaavaa ohjelmaa ei ollut muualta saatavilla.

Ohjelma toteutettiin C#-ohjelmointikielellä käyttäen apuna .NET frameworkia. Työssä käytetty ohjelmointiympäristö on Microsoftin Visual Studio 2013. Tietojen tallentaminen tehtiin XML-tiedostoon.

Opinnäytetyön kirjallinen osuus alkaa rakennusautomaatiojärjestelmien ja käyttöliittymäteorian esittelyllä. Tämän jälkeen esitellään ohjelman kehitykseen käytetyt työkalut ja lopuksi ohjelma osiin purettuna.

REMOTE INTERFACE APPLICATION FOR BUILDING AUTOMATION SYSTEMS

Vuorela, Ville
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
May 2015
Supervisor: Trast, Ismo
Number of pages: 34
Appendices: 0

Keywords: Building automation, Control room, User interface, C#

The following thesis looks over the developing process of a program made in spring 2015. The program was made to ease the work of control room employee in building automation system's control center by giving the user a way to open a remote control session to predefined targets by clicking buttons on a map template. In addition the user had to have a possibility to add, remove and modify existing targets. There was no equivalent program available.

The software was developed using C# programming language and .NET framework. The IDE used was Microsoft's Visual Studio 2013. The data saving was done into an XML document.

The report of this thesis begins by introducing the reader to building automation systems and user interfaces. The next part of this report explains the tools used in development of the program. At the end of the report the developed software is presented in smaller parts.

LYHENTEET

CLI	Command Line Interface
DLL	Dynamic-Link Library
EXE	Executable
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
JIT	Just-In-Time
PC	Personal Computer
TUI	Text-based User Interface
VNC	Virtual Network Computing
XML	Extensible Markup Language

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön toimeksiantaja on Puolustushallinnon rakennuslaitos. Opinnäytetyön tavoitteena on kehittää Windows-käyttöjärjestelmällä toimiva apuohjelma helpottamaan valvomotyöntekijän työtä kiinteistöautomaatiojärjestelmän keskusvalvomossa. Ohjelmointikielenä työssä käytetään C#:ia ja käytössä oleva ohjelmointiympäristö on Microsoftin Visual Studio 2013.

Tässä raportissa esitellään ensin ympäristö, jonne ohjelma kehitetään. Tämän jälkeen kerrotaan perusteita rakennusautomaatiojärjestelmien toiminnasta ja teoriaa käyttöliittymäsuunnittelusta. Raportin viimeisessä osassa esitellään ohjelman suunnitteluun käytettäviä työkaluja sekä kehitettävä ohjelma osa kerrallaan.

Kehitettävän ohjelman tavoite on nopeuttaa valvomotyöntekijän työtä hälytyksen tullessa antamalla hänelle keino käynnistää etäyhteys haluttuun kohteeseen nappia painamalla. Käyttäjän on voitava itse konfiguroida ohjelmaa, eli hänen on pystyttävä lisäämään, poistamaan ja muokkaamaan kohteita.

2 KIVA 24/7 -HANKE

Valvomo, jota varten ohjelma kehitetään, on osa Puolustushallinnon rakennuslaitoksen valtakunnallista kiinteistöautomaatiovalvomohanketta KIVA 24/7. Valmistumassa oleva valvomo tulee olemaan kiinteistöautomaatiojärjestelmän uusi keskusvalvomo. Valvomoon kaivattiin uudistusta sen tämänhetkiselle valvomokäyttöliittymälle, jonka koettiin olevan epäkäytännöllinen, koska se aiheuttaa liikaa turhia vaiheita hälytyksen saannin sekä sen tarkastamisen välille.

Nykyinen valvomon toimintamalli on seuraavanlainen:

1. Valvomotyöntekijä saa Care-ohjelmaan automaattisesti hälytyksen esimerkiksi laiteviasta. Hälytys voi sijaita missä tahansa kyseiseen järjestelmään liitetyssä kiinteistössä, mahdollisesti toisella puolella Suomea.

2. Työntekijä etsii ja ajaa työpöydältä oikean, kyseiselle kohteelle konfiguroidun tiedoston, joka käynnistää TightVNC-ohjelmalla etäyhteyden kyseisen kiinteistövalvomon tietokoneelle.
3. Etäyhteyden avulla työntekijä pääsee tarkastamaan hälytyksen syyn, jonka jälkeen hän päättää millaisia jatkotoimenpiteitä mahdollisesti tarvitaan.

2.1.1 Care

Care on Caverion Oyj:n kehittämä ohjelma, jota käytetään kiinteistöautomaatiovalvonnassa hälytysten keräämiseen. Se lataa hälytykset halutulta hälytyspalvelimelta ja ilmoittaa käyttäjälle uuden hälytyksen tullessa. Caren avulla käyttäjä voi myös jakaa ilmoituksen eteenpäin sähköpostilla sekä tekstiviestillä.

2.1.2 TightVNC

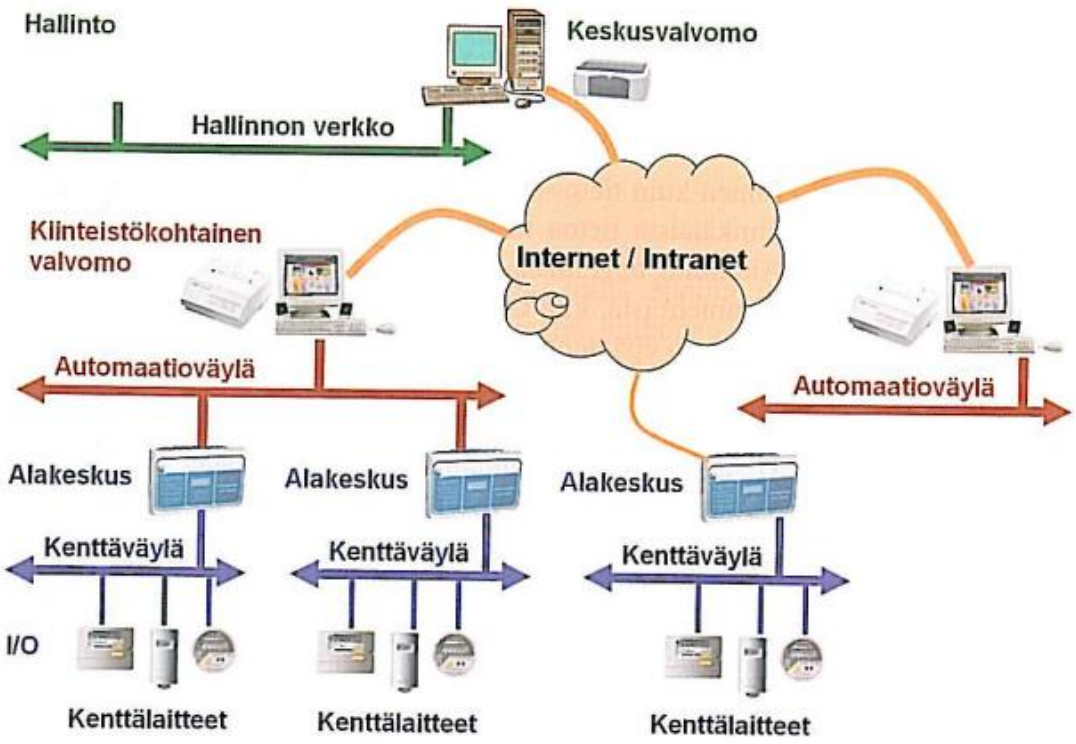
TightVNC on avoimeen lähdekoodiin perustuva etäkäyttöohjelma, joka toimii Windows- sekä Linux-käyttöjärjestelmillä. Nimensä mukaisesti se käyttää VNC-protokollaa (Virtual Network Computing). TightVNC on yhteensopiva myös muiden samaa protokollaa käyttävien ohjelmien kanssa.

3 RAKENNUSAUTOMAATIOJÄRJESTELMÄT

3.1 Rakennusautomaatiojärjestelmä

Rakennusautomaatiojärjestelmät tarkoittavat järjestelmiä, joilla voidaan automatisoida sekä etäkäyttää kiinteistöjen teknisiä toimintoja ja laitteita, kuten esimerkiksi ilmanvaihtojärjestelmiä ja valaistusten ohjausta.

Nykyiset rakennusautomaatiojärjestelmät voidaan yleensä tehtäväalueidensa perusteella jakaa kolmeen tasoon, jotka kommunikoivat toistensa kanssa. Näitä ovat kenttälaitteet, alakeskukset sekä valvomolaitteet. Kuten alla olevasta kuvasta näkyy, jaetaan valvomolaitteet usein vielä kahteen osaan: keskusvalvomoon sekä kiinteistökohtaisiin valvomoihin. (ST-Käsikirja 22 2008, 9, 12)



Kuva 1. Rakennusautomaation hierarkia (ST-Käsikirja 22 2008, 12)

3.1.1 Kenttälaitteet

Kenttälaitteet ovat rakennusautomaatiojärjestelmien alin taso. Ne tarkoittavat erilaisia mittalaitteita sekä toimilaitteita. Nämä laitteet ovat esimerkiksi erilaisia mittalaitteistoja, kuten lämpötila-, paine- ja kosteusantureita. Esimerkkinä toimilaitteista taas ovat erilaiset prosesseja säätävät venttiilimoottorit ja ilmanvaihtojärjestelmien taajuusmuuntajat, joita käytetään kierrosnopeuden säätämiseen. (ST-Käsikirja 22 2008, 12)

3.1.2 Alakeskukset

Alakeskukset ovat seuraava taso ylöspäin kenttälaitteista. Alakeskukset yhdistyvät kenttälaitteisiin kenttäväylillä. Alakeskuksista lähetetään toimintokäskyt toimilaitteille sekä siellä kerätään mittalaitteilla mitatut arvot. Tämän lisäksi alakeskukset käsittelevät mittalaitteilta tulevat viestit ja muuttavat ne fysikaalisiksi suureiksi. (ST-Käsikirja 22 2008, 12)

3.1.3 Valvomolaitteet

Valvomolaitteistot koostuvat kiinteistökohtaisista valvomoista sekä yleensä yhdestä keskusvalvomosta. Kiinteistökohtaiset valvomot ovat hierarkisesti alakeskuksien yläpuolella. Kiinteistökohtaiset valvomot ovat yhteydessä alakeskuksiin automaatioväylillä. Mikäli kyseessä on suuri rakennusautomaatiojärjestelmä, joka kattaa useita eri kiinteistöjä, voi kiinteistökohtaisia valvomoita olla isojakin määriä. Keskusvalvomosta käsin valvomotyöntekijät voivat käyttää kiinteistökohtaisia valvomoita joko internetin tai intranetin ylitse. (ST-Käsikirja 22 2008, 13)

Valvomolaitteittoistot ovat rakennusautomaatiojärjestelmässä se osa joka toimii rajapintana ihmisen ja järjestelmän välillä. Valvomoissa alempien tasojen mittaukset esitetään usein graafisesti sekä niissä mahdollistetaan toimilaitteiden manuaalinen ohjaaminen. (ST-Käsikirja 22 2008, 13)

3.2 Tavoitteet ja edut

Rakennusautomaatiojärjestelmien perimmäinen tavoite on helpottaa kiinteistöjen hoitoa automatisoimalla tehtäviä. Tämä säästää ylläpitokustannuksissa, joten se on hyvin suunniteltuna ja toteutettuna taloudellisesti kannattavaa.

Energian säästö on nykyisen vihreän ajattelumallin vuoksi hyvin tärkeä asia nyky-yhteiskunnassa. Rakennusautomaatiojärjestelmät auttavat tämän tavoitteen saavuttamisessa, koska niiden avulla voidaan hallita rakennusten energiankulutuksia siten, että asetettuihin energiankulutustavoitteisiin päästään. Tämä tapahtuu mm.

lämpötilan, ilmavirran ja valaistuksen ohjauksella silloisten tarpeiden mukaan. (Baff 2005)

Toinen rakennusautomaatiolla saavutettava suuri hyöty liittyy huolto- ja kunnossapitotoimintaan. Niitä voidaan tehostaa asetettujen tavoitteiden mukaisesti. Esimerkiksi työn seurannasta saadaan tehokkaampaa automaattisilla työraporteilla. Laitteiden automatisoitu kunnonvalvonta kohdistaa huolto- ja korjaustoimenpiteet vikaantuneisiin laitteisiin, jolloin turhien määräaikaishuoltoihin perustuvien huoltokäyntien tarve vähenee. Tätä edesauttaa erityisesti päivystystehtävien hoito etäohjauksen avulla. (Baff 2005)

Kustannussäästöt mahdollistuvat eri kulutusten tarkoilla seurannoilla ja niiden monipuolisilla raportointimahdollisuuksilla. Jos järjestelmään kuuluu useita kiinteistöjä niin niiden kulutusten vertailu onnistuu helposti kattavien raporttien avulla. (Baff 2005)

Keskitetyt valvonta- ja ohjauspaikat helpottavat hälytyskäsittelyä, mikä on yksi kiinteistönhoidon keskeisimpiä rutiineja. Hyvällä valvonnalla saadaan kiinteistön toiminnasta tarvittavat tiedot sekä reaaliaikaisena, että historiatietoihin perustuvina. (Baff 2005)

3.3 Kiinteistöautomaatiovalvomo

Kiinteistöautomaatiovalvomot koostuvat sovellustensa puolesta yleensä joko kahdesta eri tarkoitukseen kehitetystä ohjelmistosta tai yhdestä joka sisältää molemmat. Ensimmäisellä ohjelmistolla valvomotyöntekijä näkee järjestelmässä ilmenneet virheet ja hälytykset joko reaaliaikaisesti tai sitten hälytyshistoriaa apuna käyttäen. Toista ohjelmaa käytetään jotta voidaan hallita alakeskuksia sekä hierarkisesti niiden alapuolella olevia kenttälaitteita.

3.3.1 Hälytykset ja niiden käsittely

Rakennusautomaatiojärjestelmään liitetään aina kaikkien rakennusten eri järjestelmien hälytykset. Tämä tapahtuu siihen tarkoitukseen kehitetyllä ohjelmistolla, joka tarkastaa tasaisin väliajoin uudet hälytykset tätä varten pystytetyltä palvelimelta. Palvelimelle hälytykset tallentuvat automaattisesti. Kun ohjelma havaitsee uuden hälytyksen tallentuneen palvelimelle, se tulostaa näytölle huomautuksen erilliseen hälytysikkunaan. Hälytykset jäävät myös lokitietoihin jotta niitä voidaan tutkia jälkikäteen. (ST-Käsikirja 22 2008, 51)

Hälytykset voidaan jaotella kahdella eri tavalla.
Kiireellisyyden perusteella jaetut hälytykset:

- A-luokan hälytykset ovat turvallisuushälytyksiä. Esimerkiksi palo-, rikos ja hissihälytykset.
- B-luokan hälytykset ovat yleensä kiireellisiä hälytyksiä, kuten esimerkiksi jäähdytyskoneiden ja kylmiöiden viat.
- C-luokan hälytykset ovat kiireettömiä hälytyksiä. Yleensä niitä ovat prosessien ja erillispisteiden hälytykset.
- D-luokan hälytykset ovat yleisiä huoltohälytyksiä, jotka ovat esimerkiksi muistutuksia käyttötuntimääriin perustuvista huoltokäynneistä.

(ST-Käsikirja 22 2008, 57-59)

Toinen tapa jakaa hälytykset on niiden jakaminen hälytysten lähteiden mukaan:

- Järjestelmän sisäiset hälytykset. Nämä kattavat rakennusautomaatiojärjestelmän omaan toimintaan liittyvät viat, kuten esimerkiksi anturiviat ja kommunikaatioviat järjestelmän eri osien välillä.
- LVI-prosessihälytykset. Näihin kuuluvat mittausten raja-arvohälytykset sekä koneiden ristiriitahälytykset.
- Erillisjärjestelmien hälytykset ovat erillisten sähkö- ja LVI-järjestelmien yksittäisiä hälytyksiä. Näistä esimerkkejä ovat sähkökatkoshälytykset sekä erilaiset turva- sekä huoltohälytykset.

(ST-Käsikirja 22 2008, 51-57)

3.3.2 Etäkäyttö

Olemassa olevia rakennusautomaatiojärjestelmiä alettiin kehittämään internetpohjaisiksi 90-luvun loppupuolella. Silloin kuitenkaan internet-liittymien nopeudet eivät vielä mahdollistaneet kunnollista etävalvontaa, koska grafiikan siirto verkon ylitse oli hidasta ja tämän vuoksi vasteajat olisivat olleet liian pitkiä. Laajakaistojen tulo on mahdollistanut etäkäytön. (ST-Käsikirja 22 2008, 146)

Kun rakennusautomaatiojärjestelmässä on useita valvomoita, on etäkäytön tarkoituksena antaa valvomotyöntekijöille mahdollisuus käyttää myös muiden valvomoiden alaisia laitteita. Etäkäytön yli työntekijä näkee toisessa valvomossa olevan PC:n ruudun sellaisenaan ja voi käyttää tätä PC:tä samalla tavalla kuin hän olisi fyysisesti sen äärellä. Tämä vähentää valvomotyöntekijöiden määrän tarvetta, sillä jokaisessa valvomossa ei välttämättä tarvita päivystäjää. (ST-Käsikirja 22 2008, 148)

Etäkäytössä on hyvä muistaa tietoturva. Ottaessa etäyhteyden valvomosta valvomoon julkisen internetin yli, on hyvä suojata liikenne salatulla VPN-tunneloinnilla. VPN-tunnelointi kryptaa sen läpi kulkevan datan, jolloin mahdollinen salakuuntelu ei paljasta kulkevan datan sisältöä, joka voi olla hyvinkin arkaluontoista. (ST-Käsikirja 22 2008, 155)

4 KÄYTTÖLIITTYMÄTEORIA

4.1 Käyttöliittymän määritelmä

Käyttöliittymillä tarkoitetaan kaikkea ihmisen ja koneen välillä olevaa rajapintaa, joka mahdollistaa tiedon siirron käyttäjän sekä laitteiden välillä. Käyttöliittymän avulla välitetään käyttäjälle tietoa käytön kohteena olevan laitteen toiminnasta, sekä sen avulla käyttäjä voi ohjata laitteiden toimintaa käyttöliittymän sallimissa rajoissa. (ST-Käsikirja 22 2008, 15)

Käyttöliittymät kuitenkin voidaan jaotella useaan eri osa-alueeseen. Seuraavaksi esitetään Ben Shneidermanin jo 1980-luvulla tekemä käyttöliittymien jaottelu osa-alueisiin hyvinkin käytännönläheisesti. (Kallio 1992, 13)

4.1.1 Vuorovaikutustyylit

Vuorovaikutustyyleillä käyttöliittymät voidaan jaotella erilaisiin tyyleihin, kuten komento-, valikko- sekä lomakepohjaisiin käyttöliittymiin. Komentopohjaiset käyttöliittymät ovat nykyään harvinaistumassa erilaisten graafisten käyttöliittymien tieltä, mutta niitäkin silti yhä käytetään. Lomakepohjaisia käyttöliittymiä käytetään tiedon syötössä, jossa täytettäviä kenttiä on useita peräkkäisiä. Valikkopohjaiset käyttöliittymät löytyvät nykyään usein esimerkiksi graafisten ohjelmien ylälaidasta, jossa voidaan navigoida valikoiden ja niiden alivalikoiden avulla ohjelman eri osiin. (Kallio 1992, 13, 28)

4.1.2 Tiedon syöttötekniikat

Esimerkkejä erilaisista syöttötekniikoista ovat tavallisten hiirien sekä näppäimistöjen lisäksi myös haptiset, eli tuntoaistiin perustuvat, sekä äänellä ohjattavat käyttöliittymät. (Kallio 1992, 13)

4.1.3 Näyttöjen layout ja sisältö

Näyttöjen organisoinnissa on kyse kaikesta siitä, mitä näytölle sijoitetaan ja mihin kohtaan. Tässä työssä keskitytään pääasiassa valvomokäyttöliittymien tähän osa-alueeseen. (Kallio 1992, 13)

4.1.4 Vasteaika

Vasteaika tarkoittaa, viivettä mikä on käyttäjän antaman syötteen sekä tietokoneen antaman palautteen välissä. Tämä luonnollisesti yritetään yleensä minimoida parhaimman käyttökokemuksen saavuttamiseksi. (Kallio 1992, 13)

4.1.5 Virheiden käsittely

Virheiden käsittelyssä yritetään eliminoida käyttäjien tekemät virheet. Eritoten vakavien virheiden välttäminen on tärkeää, jotta ohjelmat eivät kaadu. (Kallio 1992, 13)

4.1.6 Yksilölliset erot

Yksilöillä tai persoonilla tarkoitetaan erilaisten käyttäjien merkitystä käyttöliittymän suunnittelun kannalta. Sovellusta suunnitellessa on aivan aluksi määriteltävä kenelle ohjelmaa suunnitellaan. Teknisen koulutustaustan omaava ja paljon erilaista tietotekniikkaa käyttänyt henkilö ajattelee usein ohjelman käyttöä eri tavalla verrattuna ihmiseen joka ei ole tietokoneohjelmia niin paljoa käyttänyt.
(Kallio 1992, 13, 49)

4.2 Sovellusten käyttöliittymät

Toiminnallisuuden perusteella käyttöliittymät voidaan yleensä jakaa kahteen pääryhmään: Komentoliittymiin (Command Line Interface, CLI) sekä graafisiin käyttöliittymiin (Graphical User Interface, GUI).

4.2.1 Komentoliittymä

Komentoliittymä on perinteinen käyttöliittymän muoto, joka kehittyi reikäkortteihin perustuneen käyttöliittymän tilalle. Komentoliittymässä käskyt annetaan eksaktissa kirjoitetussa muodossa, joka tarkoittaa sitä että voidakseen käyttää ohjelmaa tehokkaasti, pitää käyttäjän opetella kaikki tarvitsemansa komennot ulkoa. Tämän vuoksi komentoliittymiin perustuvien ohjelmien käytön opettelu on usein haastavaa ja aikaa vievää, eikä niitä kovin usein käytetä tavallisille loppukäyttäjille tarkoitetuissa ohjelmissa.

Komentoliittymissä on kuitenkin muutamia etuja graafisiin käyttöliittymiin verrattuna. Opittuaan komennot voi käyttäjä tehdä monimutkaisiakin toimintoja vain yhdellä kirjoitetulla tekstirivillä, jonka tekeminen graafisen käyttöliittymän avulla voisi vaatia paljonkin navigointia eri elementtien välillä. Komentoliittymiin perustuvat ohjelmat myös vievät paljon vähemmän resursseja tietokoneelta johtuen piirretyn grafiikan puutteesta.

```
[vnctest@thinq2 ~]$ ls -l
total 32
drwxr-xr-x. 2 vnctest vnctest 4096 Apr 13 22:27 Desktop
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Documents
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Downloads
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Music
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Pictures
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Public
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Templates
drwxr-xr-x. 2 vnctest vnctest 4096 Mar 13 16:11 Videos
```

Kuva 2. Käyttäjän kotihakemisto ls-komentorivikomennolla listattuna

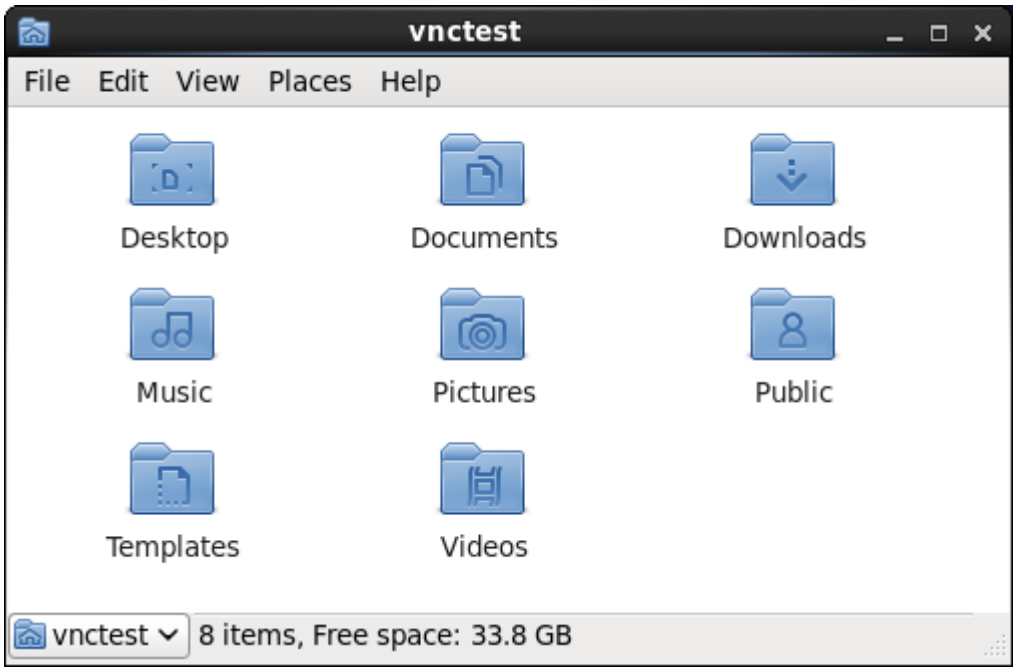
4.2.2 Graafinen käyttöliittymä

Graafinen käyttöliittymä on komentoliittymään verrattuna hyvin nykyaikainen käyttöliittymän muoto. Graafinen käyttöliittymä kehittyi komentoliittymistä tekstipohjaisen käyttöliittymän (Text-based User Interface, TUI) kautta. Yksi ensimmäisistä graafista käyttöliittymää käyttävistä tietokoneista oli Xerox PARC:in kehittämä Alto-tietokone, jonka patentoimatta jättämisen vuoksi mm. Microsoft ja

Apple myöhemmin käyttivät samanlaista käyttöliittymää omissa tuotteissaan. (Wikipedia: History of the graphical user interface 2015)

Graafinen käyttöliittymä perustuu tekstin sijaan näytölle sijoitettaviin graafisiin elementteihin ja ikoneihin. Tämä mahdollistaa hiiren käytön tietokoneita käyttäessä. Komentoliittymään verrattuna graafisessa käyttöliittymässä on ainakin kaksi isoa etua: Käytön oppimisen helppous ja paremmat mahdollisuudet usean asian tekemiseen samaan aikaan.

Tässä opinnäytetyössä kehitettävä ohjelma tulee käyttämään graafista käyttöliittymää.



Kuva 3. Käyttäjän kotihakemisto avattuna graafisella ohjelmalla

4.3 Käyttöliittymät kiinteistövalvonnassa

Kiinteistövalvonnassa käyttöliittymien perustarkoituksena on kertoa käyttäjälle kaikki tämän tarvitsema tieto kiinteistön olosuhteista, tapahtumista ja teknisten prosessien tilasta. Käyttöliittymän avulla käyttäjän pitää voida seurata, valvoa sekä tarpeen mukaan ohjata järjestelmien toimintaa. (ST-Käsikirja 22 2008, 15)

Kiinteistön valvomojärjestelmissä voidaan käyttöliittymät jaotella seuraavasti laajuuden mukaan: laite- ja tilakohtaiset, osajärjestelmäkohtaiset, koko tietojärjestelmän kattavat ja useita tietojärjestelmiä yhdistävät käyttöliittymät. (ST-Käsikirja 22 2008, 15)

Laite- ja tilakohtaiset käyttöliittymät ovat kiinteistöautomaatiossa se kaikista yksinkertaisin käyttöliittymämuoto. Näihin lasketaan erilaiset ohjauspaneelit, painikkeet ja paikallisnäyttölaitteet. Laite- ja tilakohtaisten käyttöliittymien ei aina tarvitse antaa palautetta, ja jos tarvitsee, voi se olla esimerkiksi jokin hyvinkin yksinkertainen merkkivalo suoritetun toiminnon tilan muutoksesta. Monimutkaisimmillaan tämän tyyppiset käyttöliittymät voivat olla kosketusnäytöllisiä symboleilla toimivia käyttöliittymiä. (ST-Käsikirja 22 2008, 17)

Osajärjestelmät tarkoittavat tietojärjestelmissä jonkinlaisen toiminnallisen osakokonaisuuden muodostavaa osaa. Näitä ovat esimerkiksi oman käyttöliittymän sisältävät ilmastointilaitteistot ja varavoimalaitteistot. Tässä yhteydessä tietojärjestelmät tarkoittavat yhden tai useamman kiinteistön kattavia kokonaisuuksia, kuten paloilmoitinjärjestelmiä ja murtohälytysjärjestelmiä. Nämä käyttöliittymät voivat olla myös PC-laitteissa toimia ohjelmistoja. (ST-Käsikirja 22 2008, 19)

Koko tietojärjestelmän kattavat käyttöliittymät ovat usein valvomossa olevia graafisia tai tekstipohjaiseen valikkorakenteeseen perustuvia käyttöliittymiä. Nämä ovat usein PC-sovelluksia, mutta ne voivat myös olla valmistajakohtaisia järjestelmien keskuslaitteistoihin liitettyjä käyttöpaneelin kautta käytettäviä käyttöliittymiä. (ST-Käsikirja 22 2008, 20)

Useita tietojärjestelmiä yhdistävät käyttöliittymät sijaitsevat usein keskusvalvomoissa, jotka kattavat rakennusautomaation tapauksessa monia kiinteistökohtaisia valvomoita. Nämä käyttöliittymät ovat pääasiassa PC-sovelluksia, joihin integroidaan useamman tietojärjestelmän käyttöliittymäohjelmistoja joko niin että ne toimivat toisista riippumattomina, tai siten että eri tietojärjestelmät tuottavat tiedon yhteiseen ohjelmistoon. Jälkimmäisen hyötynä on se, että sillä voidaan esittää samanaikaisesti tietoa useasta eri tietojärjestelmästä. Operointiperiaatteet useiden ja

yksittäisten tietojärjestelmien käyttöliittymissä eivät eroa toisistaan. (ST-Käsikirja 22 2008, 21-22)

4.4 Käyttöliittymän suunnittelu

Tietokonesovellusten kehityksen alkuaikoina kehittäjät kehittivät ohjelmat pääasiassa itselleen ja muille teknisen koulutustaustan omaaville henkilöille. Tämä aiheutti usein monimutkaisia, mutta kuitenkin toimivia käyttöliittymäratkaisuja, joita tavallisen käyttäjän oli hankala omaksua. Nykyään kun käyttäjät ovat monipuolistuneet, ei sovelluksen kehittäjä voi luottaa ainoastaan omaan intuitioonsa käyttöliittymää suunnitellessa. Kehittäjien, jotka seuraavat tarkkaan kehitysvaiheessa olevien ohjelmiensa käyttäjiä ja niiden käyttötapoja samalla muokaten havaintojensa perusteella käyttöliittymää, todetaan usein tuottavan korkealaatuisia käyttöliittymiä. (Shneiderman & Plaisant 2010, 116)

Onnistuneen käyttöliittymän suunnittelu voidaan jakaa neljän peruspilarin varaan. Nämä ovat käyttöliittymän vaatimukset, ohjesääntödokumentit, prototyypit ja käytettävyyden testaus. Näillä neljällä pilarilla saadaan hyvät ideat muutettua onnistuneiksi toteutuksiksi. (Shneiderman & Plaisant 2010, 120)

4.4.1 Käyttöliittymän vaatimukset

Käyttäjien tarpeiden tarkka määrittely on olennaisessa osassa kaikissa hyvin onnistuneissa suunnitteluprosesseissa. Käyttöliittymäsuunnittelussa käyttöliittymien vaatimusten määrittely vaihtelee eri alojen välillä, mutta kuitenkin haettu lopputulos on aina sama: selvä määritelmä käyttäjistä sekä tehtävistä joita ohjelmalla tullaan tekemään. Määritelmissä pitää selvitä sekä järjestelmän vaatimukset (laitteet, ohjelmat, vakaus, suorituskyky) että myös suoraan käyttöliittymiin liittyvät vaatimukset (I/O-laitteet, käyttäjät, toiminnallisuudet). (Shneiderman & Plaisant 2010, 121)

4.4.2 Ohjesääntödokumentit

Käyttöliittymäsuunnittelun aikaisessa vaiheessa pitää luoda hyvät ohjesäännöt, jotta ohjelman käyttöliittymästä saadaan yhtenäinen kokonaisuus. Jokaisella projektilla on erilaiset tarpeet, mutta ohjesäännöissä on hyvä käsitellä seuraavia asioita:

- Sanat, ikonit ja grafiikat
    - Terminologia ja lyhenteet
    - Fontit
    - Ikonit ja nappulat
    - Värit ja taustat
- Näytön layout
    - Menut ja täytettävät kentät
    - Virhesanomat
    - Marginaalit
    - Skaalautuvuus erikokoisille näytöille
- I/O-laitteet
    - Hallinta eri laitteilla (hiiri, näppäimistö jne.)
    - Varoitusäänet
    - Vasteaika eri toiminnoille
    - Esteettömyysvaihtoehdot
- Toiminnallisuus
    - Hiirellä klikkaus ja veto
    - Komentojen syntaksi
    - Näppäinoikotiet
    - Kosketusnäytöllinen navigaatio
    - Virheiden käsittely ja niistä palautuminen

(Shneiderman & Plaisant 2010, 123)

4.4.3 Prototyypit

Yksi käyttöliittymäsuunnittelun haasteista on se, ettei käyttäjillä ole välttämättä selvää kuvaa siitä minkä näköinen ohjelma tulee olemaan valmistuttuaan. Käyttäjät

eivät myöskään aina ymmärrä, millaisia vaikutuksia eri suunnittelupäätöksillä on ja kuinka haastavaa ja kallista muutosten tekeminen on jälkikäteen.

Vaikka tähän ongelmaan ei ole täydellistä ratkaisua, ongelmia voidaan välttää kehittämällä käyttöliittymästä prototyyppejä ja antamalla niitä käyttäjille testattaviksi. Helpoimmillaan esiteltävä käyttöliittymän prototyyppi voi olla paperille tulostettu kuva, mutta tietokoneella oleva ohjelma on sille paljon realistisempi vastine. Käyttöliittymän prototyyppi vastaa lopputuotetta ainoastaan ulkonäöllisesti, joten se voidaan toteuttaa verrattain helposti ilman aikaa vievää ohjelmointia. (Shneiderman & Plaisant 2010, 125)

4.4.4 Käytettävyyden testaus

Ennen kuin ohjelma voidaan luovuttaa käyttäjälle, pitää sitä testata varmistuakseen sen kunnollisesta toimivuudesta. Ohjelman kehittäjä ei kuitenkaan voi tehdä testausta itse, koska yleensä tekijä tulee sokeaksi omille virheilleen. Vaikka prototyyppien testautus kollegoilla tai käyttäjillä antaa hyvää palautetta, ovat testausta työkseen tekevät asiantuntijat parempi vaihtoehto. Asiantuntija pitää ensin perehdyttää ohjelman toimintaympäristöön ja sen toimintamalleihin. Testauksia kannattaa suunnitella projektin eri vaiheisiin ja testaavia asiantuntijoita kannattaa olla useita jotta suurin osa ongelmista huomattaisiin ennen ohjelman päätymistä loppukäyttäjän käsiin. (Shneiderman & Plaisant 2010, 126)

5 OHJELMAN TOTEUTUS

5.1 Toteutettava ohjelma

Päättötyön tavoitteena oli omatoimisesti suunnitella ja kehittää työkalu kiinteistöautomaatiovalvomoon helpottamaan valvomotyöntekijän työtä. Koska ohjelman suunnittelu sekä kehitys tapahtuivat pääasiassa omatoimisesti, kävin aluksi tutustumassa Jyväskylässä sijaitsevaan keskusvalvomoon. Tällä tutustumiskäynnillä minulle selvisi heidän haluamansa ohjelman tarpeet ja vaatimukset.

Ohjelman tehtävänä on käynnistää etäkäyttöohjelma TightVNC haluttuun kohdetietokoneeseen mahdollisimman nopeasti ja selkeästi. Koska kohdetietokoneita on useita ja ne sijoittuvat maantieteellisesti useisiin eri paikkoihin, haluttiin ohjelman taustalle kartta, jonka päälle etäkäyttöyhteyden käynnistävät napit sijoitetaan.

Etäkäytettävien kohdetietokoneiden muuttuvuuden vuoksi, esim. vaihtuva IP-osoite tai uusi kohdetietokone, piti ohjelmassa olevat kohteet suunnitella alusta alkaen muokattaviksi. Tämän vuoksi ohjelman teossa piti panostaa käyttöliittymään, millä valvomotyöntekijät voivat tarpeen vaatiessa luoda uusia, poistaa vanhoja tai muokata nykyisiä kohteita.

Toimintaympäristöstä johtuen ohjelma tuli luoda Windows 7 –käyttöjärjestelmällä toimivaksi.

5.2 Työkalut

Ohjelman tarpeiden vuoksi valitsin sen kehittämiseen käyttämäkseni ohjelmointikieleksi C#:in. Tämän apuna käytin kyseisen ohjelmointikielen kanssa hyvin yhteensopivaa .NET frameworkia.

C#:in ja .NET:in kanssa yhteensopivia ohjelmointiympäristöjä on muutamia, joista kolme suosittua ovat Microsoftin kaupallinen Visual Studio sekä vapaaseen

lähdekoodiin perustuvat SharpDevelop ja MonoDevelop. Käyttämäkseni ohjelmointiympäristöksi valitsin Visual Studio 2013:n.

5.2.1 C#

C#-kielen kehitys alkoi 1990-luvun lopulla Microsoftilla työskennelleen Anders Hejlsbergin vetämän tiimin voimin. Tavoitteena oli luoda Javan kaltainen moderni, helppokäyttöinen, olio-orientoitunut ja tyyppiturvallinen ohjelmointikieli, jota tullaan tarvitsemaan silloin vielä julkaisemattoman .NET alustan kanssa. Nimensä mukaisesti C#:in juuret ovat C-perheessä, eli se muistuttaa huomattavasti C:tä ja C++:aa. Näiden lisäksi sillä on suuria yhtäläisyyksiä Javan kanssa. (Hejlsberg, Wiltamuth & Golde 2006, 3)

C# on olio-orientoitunut ohjelmointikieli, jonka lisäksi se tukee komponenttisuuntautunutta ohjelmointia. Komponentit ovat itsenäisiä paketteja, joita käytetään nykyaikana ohjelmoinnissa kasvavin määrin. (Hejlsberg ym. 2006, 3)

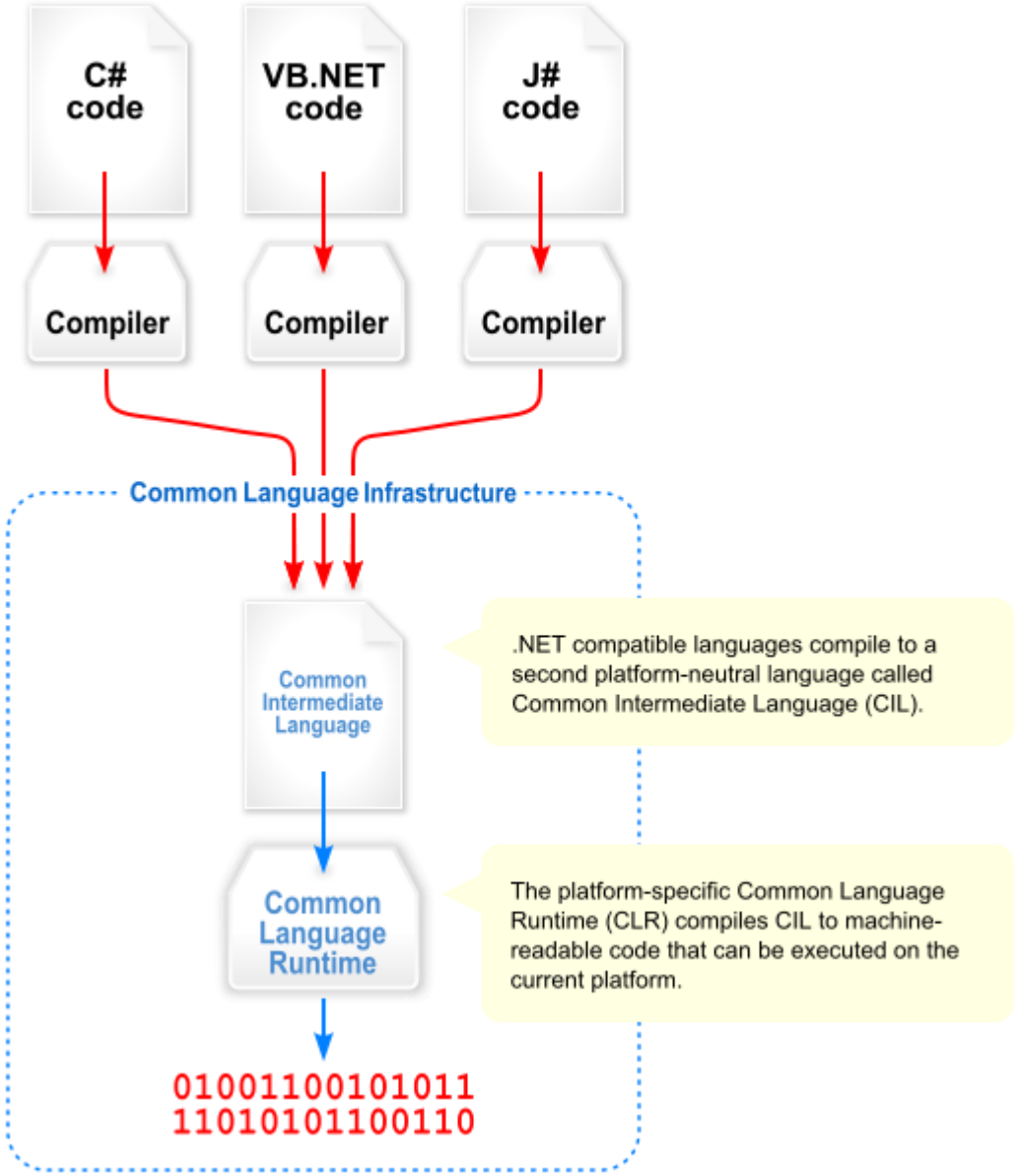
Yksi seikka, mihin C#:n suunnittelussa erityisesti keskityttiin, on versiointi (versioning). Versionnin vuoksi vanhakin koodi toimii halutunlaisesti kun käytettyjä ohjelmakirjastoja (libraries) päivitetään. Yksi käytännön esimerkki versionnista on erilliset *virtual* ja *override* modifikaattorit metodien yhteydessä. (Hejlsberg ym. 2006, 3)

5.2.2 .NET framework

Microsoftin luoma .NET framework on ajoympäristö erilaisille Windows-ohjelmille. Se kehitettiin samaan aikaan yhdessä C#:n kanssa, jonka vuoksi ne toimivat erinomaisesti yhteen. Koska C#:ssa ei ole itsessään omia kirjastoja, se hyödyntää pääasiassa .NET –kirjastoja.

On olemassa yleinen harhaluulo, että .NET käyttää samanlaista virtuaaliympäristöä ohjelmiensa ajamiseen kuin Java. Todellisuudessa koodi kuitenkin käännetään samantapaisesti kuten esimerkiksi C-ohjelmassa. Isoimpana erona C-kielestä

kääntämiseen on koodin kääntäminen käytännössä kahteen kertaan. Ensiksi koodi käännetään alustaneutraalille Common Intermediate Language:lle exe- tai dll-tiedostoksi. Tämän jälkeen ajon yhteydessä tehdään vielä ns. JIT (just-in-time) käännös, joka muuntaa sen ajettavaksi konekieleksi. (Powell & Weeks 2001, 6-7)



Kuva 4. .NET toimintaperiaate

5.2.3 Visual Studio 2013

Visual Studio 2013 on uusin julkaistu versio Microsoftin paljon käytetystä ohjelmointiympäristöstä. Sisäänrakennettuna se tukee useita kieliä, kuten C:tä,

C++:aa, C#:ia ja Visual Basic .NET:iä. Asentamalla siihen erillisiä paketteja, se saadaan tukemaan myös monia muita kieliä.

Visual Studiosta, kuten monesta muustakin ohjelmointiympäristöstä, löytyy monia ohjelmointia helpottavia ja nopeuttavia tekijöitä. Alla lista mielestäni tärkeimmistä Visual Studion ominaisuuksista.

- Sisäänrakennettu kääntäjä – Nopeuttaa ohjelman testaamista, koska se mahdollistaa ohjelman ajamisen nopeasti uudelleen koodin muuttamisen jälkeen.
- Koodin täydentäminen – Visual Studiossa tämä työkalu on nimeltään IntelliSense. Koodia kirjoittaessa ohjelma antaa ehdotuksia esimerkiksi olemassa olevien muuttujien ja metodien nimistä.
- Refaktorointi – Koodin muokkaaminen ilman sen vaikutusta toiminnallisuuteen. Mahdollistaa esimerkiksi muuttujan tai luokan nimen helpon vaihtamisen koko lähdekoodissa.
- Ohjelman ulkonäön suunnitteluun tarkoitettu graafinen työkalu – Tämän avulla ohjelman suunnittelija voi helposti sijoittaa käyttöliittymän graafisia osia haluamilleen paikoille.
- Debugger – Ohjelman suorittamisen tilassa, jossa keskitytään virheiden etsintään. Mahdollistaa esimerkiksi ohjelman ajamisen halutussa kohdassa rivi kerrallaan. Helpottaa huomattavasti koodissa olevien semanttisten virheiden etsintää.
- Syntaksin tarkastaminen – Huomauttaa ohjelmoijaa lähdekoodin syntaksivirheistä jo ennen kääntäjää.
- Koodipätkien automaattinen generointi – Voidaan luoda helposti ja nopeasti pohjia esimerkiksi uudelle luokalle.

5.3 Ohjelman kehitys

Ohjelman kehitys tapahtui ketterän ohjelmistokehityksen (agile development) menetelmiä mukaillen. Koska kehitys tapahtui muutamaa tapaamista

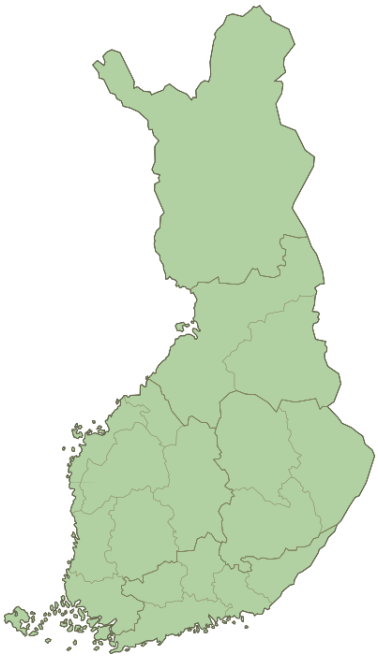
lukuunottamatta omatoimisesti, ei kehitysprosessia voida täysin ketteräksi sanoa. Kuitenkin se noudatti muutamia ketterän ohjelmistokehityksen perusperiaatteita, kuten jokaisen osa-alueen tai toiminnon valmistumista ennen seuraavaan siirtymistä sekä jatkuvaa ohjelman testausta läpi kehitysprosessin.

Seuraavissa kappaleissa esittelen ohjelman kehitysprosessin jaettuna muutamaan isoon osaan.

5.3.1 Pääikkuna ja Dynaamiset napit

Ohjelmaa käyttäessä sen tärkein ikkuna on pääikkuna, koska se on näkyvissä suurimman osan ajasta. Pääikkunan käyttöliittymä koostuu yläreunassa sijaitsevasta valikosta, taustalla olevasta karttakuvasta sekä sen päälle ajon aikana sijoitettavista napeista.

Taustalle sijoitettavaksi karttakuvaksi valitsin Wikipediasta löytämäni public domainiin kuuluvan kuvan Suomen kartasta. Public domainissa oleva kuva on vapaasti käytettävissä, koska silloin tekijä on luopunut suurimmasta osasta tekijänoikeuksistaan. Kuva on valittu vertailemalla useita kuvia, ja valitsemalla näistä selkein.



Kuva 5. Valittu karttapohja

Pääikkunan toinen osa on yläreunassa sijaitseva valikko. Valikosta löytyy keino sammuttaa ohjelma, sekä ainoa tapa siirtyä ohjelman asetuksiin.

Kuva 6. Pääikkunan ylävalikot

Pääikkunan viimeinen osa oli ohjelman käytön aikana lisättävien nappien luominen. Se osoittautui ensimmäiseksi ohjelmoinnilliseksi haasteeksi työtä tehdessä. Pienellä googlaamisella löysin Microsoft Developer Network –sivuilta siihen tarkoitukseen hyvin sopivaksi keinoksi control arrayn käytön. Control array mahdollisti nappien helpon luomisen ja poistamisen ohjelman ajon aikana sijoittamalla ne taulukkoon. Jotta napit saatiin käynnistämään etäkäyttöohjelma, lisättiin niiden event handlereiksi TightVNC:n exe-tiedoston käynnistäminen IP-osoite- ja portti-parametreilla varustettuna.

5.3.2 XML

XML (Extensible Markup Language) on syntaksiltaan HTML:n kaltainen kieli, jota voidaan käyttää tiedon tallentamiseen. Ohjelma tarvitsi tavan tallentaa tietoa pysyvästi, jotta ne olisivat tallessa vielä sen uudelleenkäynnistämisenkin jälkeen. Vertailtuani kahta siihen tarkoitukseen sopivaa tekniikkaa, XML:ää sekä SQLiteä, päädyin XML:ään.

XML-dokumenttiin tallentaminen ja sen lukeminen C#:lla käy helposti käyttämällä valmista XmlDocument-luokkaa. Tiedoston sijainti on ohjelman omassa kansiossa ja mikäli tiedostoa ei ole, ohjelma luo automaattisesti uuden. Kuvassa 7 on esimerkki XML-dokumentista johon on tallennettu TightVNC:n exe-tiedoston sijainti sekä tiedot yhdestä kohteesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  <path>C:\Program Files\TightVNC\tvnviewer.exe</path>
  <nappi>
    <name>Pori</name>
    <xcoord>101</xcoord>
    <ycoord>412</ycoord>
    <IP>192.168.1.1</IP>
    <port>100</port>
    <description>Testikohde</description>
  </nappi>
</list>
```

Kuva 7. Ohjelman luoma XML-dokumentti

5.3.3 Asetusikkuna

Ohjelmaan piti lisätä keino jolla käyttäjä voi lisätä, muokata ja poistaa kartalla olevia kohteita. Tämän lisäksi ohjelmalle pitää voida kertoa TightVNC:n exe-tiedoston sijainti, jotta ohjelma voi sen käynnistää. Näihin tarkoituksiin sopi asetusikkuna, jonne käyttäjä pääsee pääikkunan ylävalikosta.

Koska asetuksissa piti voida käsitellä useita kohteita, tein niille oman luokan jossa jokaisella oliolla on tarvittavat ominaisuudet. Jokaisen kohteen ominaisuuksia ovat nimi, kuvaus, IP-osoite, portti sekä sijainti kartalla. Asetuksiin tultaessa ohjelma lukee XML-dokumentin ja luo jokaisen siellä olevan kohteen perusteella oman olion. Tämän jälkeen oliot sijoitetaan listaan.

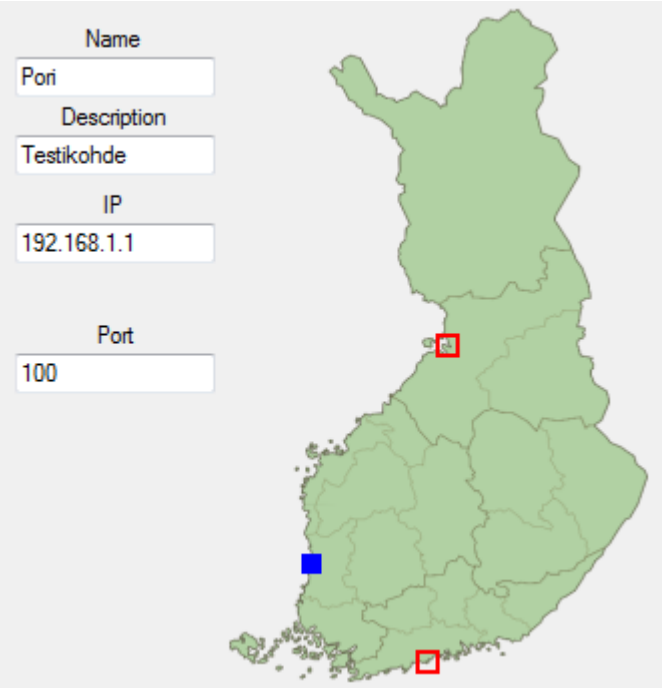
Ensimmäinen tarvittava käyttöliittymän osa oli jonkinlainen laatikko, mistä käyttäjä voi valita kohteen. Tähän tarkoitukseen sopi C#:n ListBox-luokka. Siihen sidotaan asetuksiin tultaessa luotu lista, jossa kohteet ovat. Lista näyttää jokaisella rivillä olion nimi-ominaisuuden.

Asetusikkunalle tärkeä ominaisuus on kohteiden lisääminen ja poistaminen. Tämä oli selkeyden vuoksi mielestäni hyvä sijoittaa ListBoxin läheisyyteen. Kuvassa 8 näkyy muutama ListBoxiin lisätty kohde, sekä sen alapuolella olevat lisää- ja poista-napit.

Kuva 8. ListBox sekä lisää- ja poista-napit

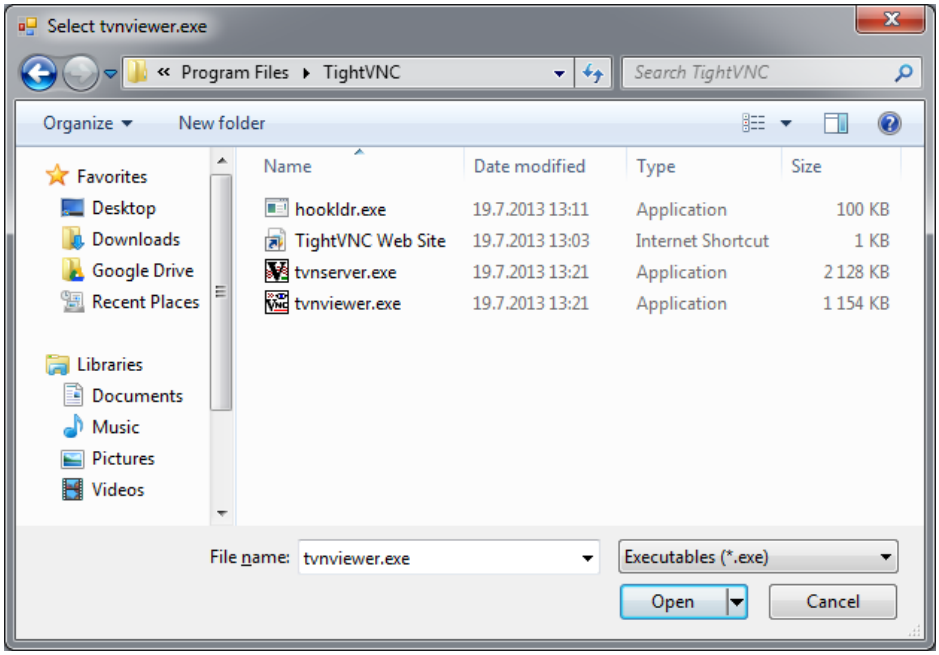
Seuraava vaihe asetusikkunan luomisessa oli kohteen ominaisuuksien muokkaamisen lisääminen. Jokaisella kohteella on neljä tekstimuotoista ominaisuutta: nimi, kuvaus, IP-osoite sekä portti. Jokaiselle niistä lisäsin oman tekstikentän, johon käyttäjä voi kirjoittaa haluamansa arvon. Tekstikenttään kirjoittaminen tallentaa sen suoraan olion tekstikenttää vastaavan ominaisuuden arvoksi. Poikkeuksena ovat IP-osoite- ja portti-kentät, jotka ensin validoivat kentän sisällön.

Viimeisenä ominaisuutena jokaisella kohteella on sen sijainti kartalla. Tätä varten lisäsin asetusikkunaan saman karttakuvan, mikä löytyy myös pääikkunasta. Valitun kohteen sijainnin voi päättää klikkaamalla kuvaa halutusta kohdasta. Kohteiden sijainti tallentuu sen suhteellisena sijaintina kuvassa. Se varmistaa nappien pysymisen oikeilla paikoilla pääikkunan kokoa muuttaessa. Kuten kuvasta 9 näkee, merkitsee ohjelma valitun kohteen sinisellä ja muut olemassa olevat kohteet punaisella.



Kuva 9. Kohteen muokkaaminen

Nyt asetusikkunasta puuttui enää ominaisuus, jonka avulla käyttäjä voi kertoa ohjelmalle TightVNC:n exe-tiedoston sijainnin. Tähän soveltui hyvin Windows-käyttäjille tuttu FileDialog-ikkuna. Kyseiseen ikkunaan päästään asetusikkunasta helposti nappia painamalla. Koska ikkunalla etsitään ainoastaan exe-tiedosto, asetin sen näyttämään oletuksena ainoastaan .exe-päätteiset tiedostot.

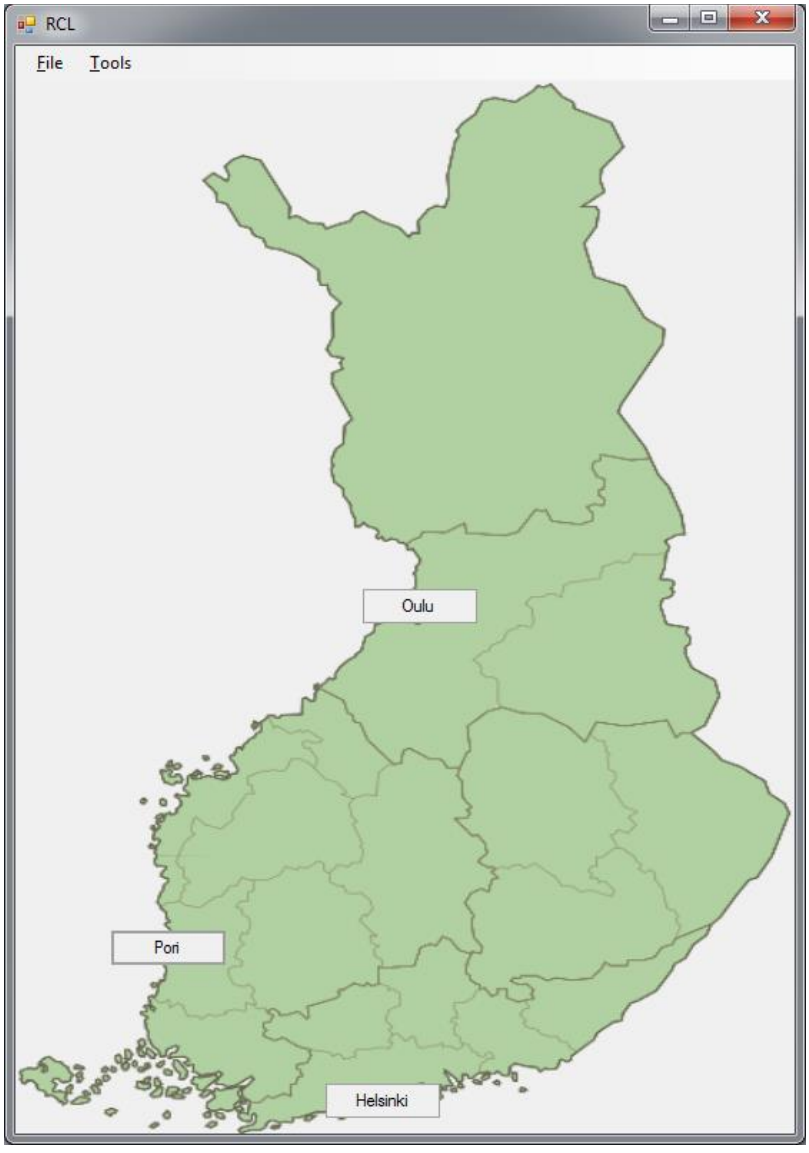


Kuva 10. FileDialog-ikkuna

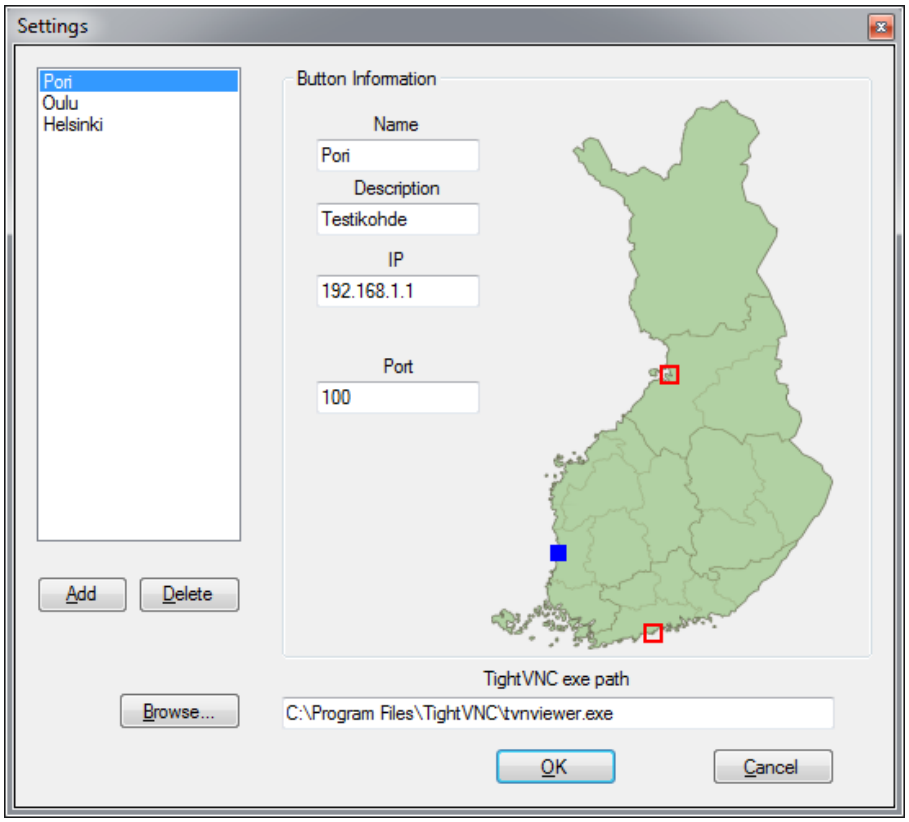
Koska asetuksista pitää päästä myös pois, lisäsin sinne Ok- ja Cancel-napit. Ok:n painaminen kirjoittaa kaikki asetukset XML-dokumenttiin ja sulkee ikkunan. Cancelin painaminen sulkee ikkunan tallentamatta.

5.4 Lopputulos

Projektin lopputuloksena on toimiva ohjelma, jonka avulla valvomotyöntekijä saa helposti käynnistettyä etäyhteyden haluamaansa tietokoneeseen. Nimeksi ohjelmalle annoin RCL, joka on lyhenne sanoista Remote Connection Launcher.



Kuva 11. Pääikkuna



Kuva 12. Asetukset

5.5 Jatkokehitysmahdollisuudet

Esiteltyäni ohjelman projektin loppuvaiheilla sain palautteena muutamia jatkokehitysmahdollisuuksia:

1. Joissakin maantieteellisissä pisteissä on useita etäkäytettäviä tietokoneita. Tämä aiheuttaa ahtautta kartalle, joten ne pitäisi mielellään saada lisättyä yhden napin taakse. Tämän toteuttaminen toisi todennäköisesti suuria muutoksia asetusikkunan käyttöliittymään, mutta se olisi kuitenkin toteutettavissa.
2. Ohjelmaan haluttaisiin mahdollisesti mahdollisuus TightVNC:n lisäksi myös muiden ohjelmien käynnistämiseen samalta kartalta. Tämän toteuttaminen vaatisi pieniä muutoksia koko työhön, mutta se olisi kuitenkin helposti toteutettavissa.

3. Koska ohjelma on tarkoitettu käytettäväksi hälytysten tarkistamisen yhteydessä, voisi ohjelmaan lisätä ominaisuuden, joka korostaa kartalla kohteita, joista hälytykset ovat saapuneet. Tämän toteuttaminen vaatisi yhteyden muodostamisen hälytystietokantaan. Yhteyden lisäksi ohjelman pitäisi osata yhdistää hälytyksen sijainti kartalla olevaan kohteeseen.

Jatkokehitys tulee mahdollisesti tapahtumaan opinnäytetyön päättymisen jälkeen.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli omatoimisesti kehittää apuväline helpottamaan valvomotyöntekijän työtä. Ohjelman suunnitteleminen ja toteuttaminen alusta alkaen oli haastaava ja hyvin mielenkiintoinen projekti. Työ kuitenkin jäi eräänlaiseen demovaiheeseen odottamaan jatkokehitystä.

Suurimmaksi haasteeksi ohjelman itsenäisessä kehityksessä ilmeni testaajien puute. Vaikka suoranaisten bugien löytäminen pienestä ohjelmasta ei ollut kovinkaan vaikeaa, olisivat testaajat helpottaneet käytettävyyden testaamista huomattavasti. Ainoastaan testaamalla ohjelmaa tarpeeksi monipuolisesti voidaan löytää ja korjata siinä olevat ongelmakohdat ja täten tuottaa hyvälaatuinen sovellus.

Toinen työssä esiintullut haaste oli itselle aiemmin käyttämättömän ohjelmointikielen käyttö. Tästä syystä opiskelin kieltä ennen työn ohjelmointiosuuden aloittamista. Siitä huolimatta työn kuluessa tuli ilmi paljon asioita, joiden tietäminen alusta saakka olisi tehnyt ohjelmoinnista huomattavasti helpompaa.

LÄHTEET

Baff. 2005. Rakennusautomaatiolla saavutettavissa olevat hyödyt. Viitattu 22.1.2015. http://www.automaatioseura.fi/index/tiedostot/BAFF_%20hyodyt.pdf

Hejlsberg, A., Wiltamuth, S. & Golde, P. 2006. The C# Programming Language. 2. Lontoo: TBS.

Kallio, T. 1992. Käyttöliittymät. Espoo: Suomen Atk-kustannus Oy.

Powell, R. & Weeks, R. 2001. C# and the .NET Framework: The C++ Perspective. Sams Publishing.

Shneiderman, B. & Plaisant, C. 2010. Designing the User Interface. Lontoo: Pearson Education.

ST-Käsikirja 22 Kiinteistöjen valvomojärjestelmät. 2008. Sähkötieto ry.

Wikipedia: History of the graphical user interface. 2015. Viitattu 23.1.2015. http://en.wikipedia.org/wiki/History_of_the_graphical_user_interface