OULUN SEUDUN
AMMATTIKORKEAKOULU

Markus Kenttälä

BUILDSERVER-OHJELMA EB PROPSIM -RADIOKANAVAEMULAATTOREILLE

BUILDSERVER-OHJELMA EB PROPSIM -RADIOKANAVAEMULAATTOREILLE

Markus Kenttälä
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistojen kehitys

Tekijä: Markus Kenttälä
Opinnäytetyön nimi: BuildServer-ohjelma EB Propsim -radiokanavaemulaattoreille
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 35

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa ohjelma, joka toimii emulaatioiden käännöspalvelimena EB Propsim -radiokanavaemulaattoreille. Opinnäytetyön tilaajana oli Elektrobit System Test Oy, joka on Elektrobit Oyj:n tytäryhtiö. Toteutettu ohjelma tulee olemaan osa uutta EB Propsim -radiokanavaemulaattoreiden käyttöliittymää.

Ohjelma toteutettiin Microsoft Visual Studio 2005 -kehitysympäristössä Windows Forms -rajapintaa käyttäen. Ohjelmistokomponenttikirjastona käytettiin .NET Frameworkin 2.0-versiota. Käyttöliittymä toteutettiin C#-kielellä. Käyttöliittymän ja C++-kielellä toteutetun kirjaston välisen rajapinnan toteutukseen käytettiin C++/CLI-kieltä. Ohjelmaa testattiin työpisteellä jatkuvasti simuloidussa ympäristössä ja loppuvaiheessa eri EB Propsim -radiokanavaemulaattoreilla laboratoriossa.

Opinnäytetyön tuloksena saatiin kehitettyä toimiva ohjelma emulaatioiden kääntämiseen EB Propsim -radiokanavaemulaattoreille tulevaa uutta käyttöliittymää varten.

Asiasanat: C#, C++/CLI, radiokanavaemulaattorit, Windows Forms

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author: Markus Kenttälä
Title of thesis: BuildServer program for EB Propsim radio channel emulators
Supervisor: Kari Laitinen
Term and year when the thesis was submitted: Spring 2013 Pages: 35

The target of this work was to design and develop a program which works as a server for emulation building for EB Propsim radio channel emulators. The customer of this thesis was Elektrobit System Test Oy which is an affiliated company of Elektrobit Oyj. The program will be part of a new user interface for EB Propsim radio channel emulators.

The program was developed in Microsoft Visual Studio 2005 IDE using Windows Forms API. .NET Framework version 2.0 was used as a software framework. User interface was developed with the C# programming language. The interface between the user interface and library was developed with C++/CLI. The program was first tested in a simulated environment in a workstation, and later in laboratory with different EB Propsim radio channel emulators.

As a result of this work was a tested program to build emulations for EB Propsim radio channel emulators for a new upcoming user interface.

Keywords: C#, C++/CLI, radio channel emulators, Windows Forms

ALKULAUSE

Haluan kiittää Elektrobit System Test Oy:tä opinnäytetyön tekemisen mahdollistamisesta ja kaikkia työkavereita, jotka auttoivat opinnäytetyön tekemisessä. Haluan kiittää myös yliopettaja Kari Laitista opinnäytetyön ohjauksesta sekä lehtori Tuula Hopeavuorta kielenohjauksesta.

23.1.2013

Markus Kenttälä

SISÄLLYS

SANASTO

.NET Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto.
API	Application Programming Interface eli ohjelmointirajapinta.
C# (C sharp)	Microsoftin kehittämä oliopohjainen ohjelmointikieli .NET-alustaa varten.
C++/CLI	Microsoftin kehittämä ohjelmointikieli, jolla on mahdollista käyttää .NET-alustan ominaisuuksia.
Emulaatio	EB Propsim -radiokanavaemulaattoreissa suoritettava radiokanavaemulaatio.
JIRA	Australialaisen Atlassian-ohjelmistoyrityksen tekemä tehtävienhallintaohjelmisto.
Kirjasto	Kirjastolla tarkoitetaan tässä opinnäytetyössä C++-kielellä toteutettua DLL-luokkakirjastoa.
Kontrolli	Kontrollilla tarkoitetaan luokkaa, joka on periytetty System.Windows.Forms.Control-luokasta.
Lomake	Luokan System.Windows.Forms olio. Windows Forms -ohjelmat koostuvat lomakkeista.
Running View	Ohjelma, jossa suoritetaan emulaatioita. Osa EB Propsim -radiokanavaemulaattoreiden käyttöliittymää.
ShellDesktop	Ohjelma, joka huolehtii EB Propsim -radiokanavaemulaattorin käyttöliittymän ohjelmien välisestä kommunikoinnista sekä ohjelmien käynnistyksestä.

Scenario Wizard	Ohjelma, jolla tehdään emulaatioita. Osa EB Propsim -radiokanavaemulaattoreiden käyttöliittymää.
StarTeam	Ohjelmistokehityksessä käytettävä versionhallintaohjelmisto.
Tapahtuma	Toiminto, joka ilmoittaa ohjelmassa tapahtuvasta muutoksesta. Esimerkiksi ohjelmassa olevan painikkeen klikkaaminen voi laukaista tapahtuman.
Tapahtumankäsittelijä	Funktio, joka on liitetty tapahtumaan. Funktio suoritetaan, kun tapahtuma laukaistaan.
Windows Forms	Microsoft .NET Frameworkiin kuuluva graafinen ohjelmointirajapinta.

1 JOHDANTO

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa ohjelma, joka toimii emulaatioiden käännöspalvelimena EB Propsim -radiokanavaemulaattoreille. Emulaatiossa määritellään radioaaltojen etenemisympäristö järjestelmässä olevien eri langattoman tiedonsiirron laitteiden välille. Emulaation kääntämisellä tarkoitetaan tässä yhteydessä tilastollisten tai käyttäjän määrittelemien mallien kääntämistä impulssivastetiedostoiksi ja laitteisto-ohjaustiedostoiksi. Ohjelman tarkoituksena on käsitellä ja suorittaa muilta ohjelmilta tulevat käännöspyynnöt käyttäen olemassa olevia kääntäjiä ja näyttää käyttäjälle tietoja emulaation käännösprosessista. Ohjelma mahdollistaa myös usean emulaation asettamisen käännösjonoon, jolloin ohjelma kääntää automaattisesti kaikki jonossa olevat emulaatiot. Ohjelma tulee olemaan osa uutta EB Propsim -laitteille tulevaa käyttöliittymää.

Ohjelma toteutettiin Microsoft Visual Studio 2005 -kehitysympäristössä Windows Forms -rajapintaa käyttäen. Ohjelman toteutuksessa käytettiin C#- ja C++/CLI-kieliä. Microsoft Visual Studio 2005 valittiin kehitysympäristöksi, koska sitä on käytetty EB Propsim -radiokanavaemulaattoreiden käyttöliittymän osien kehitystyössä.

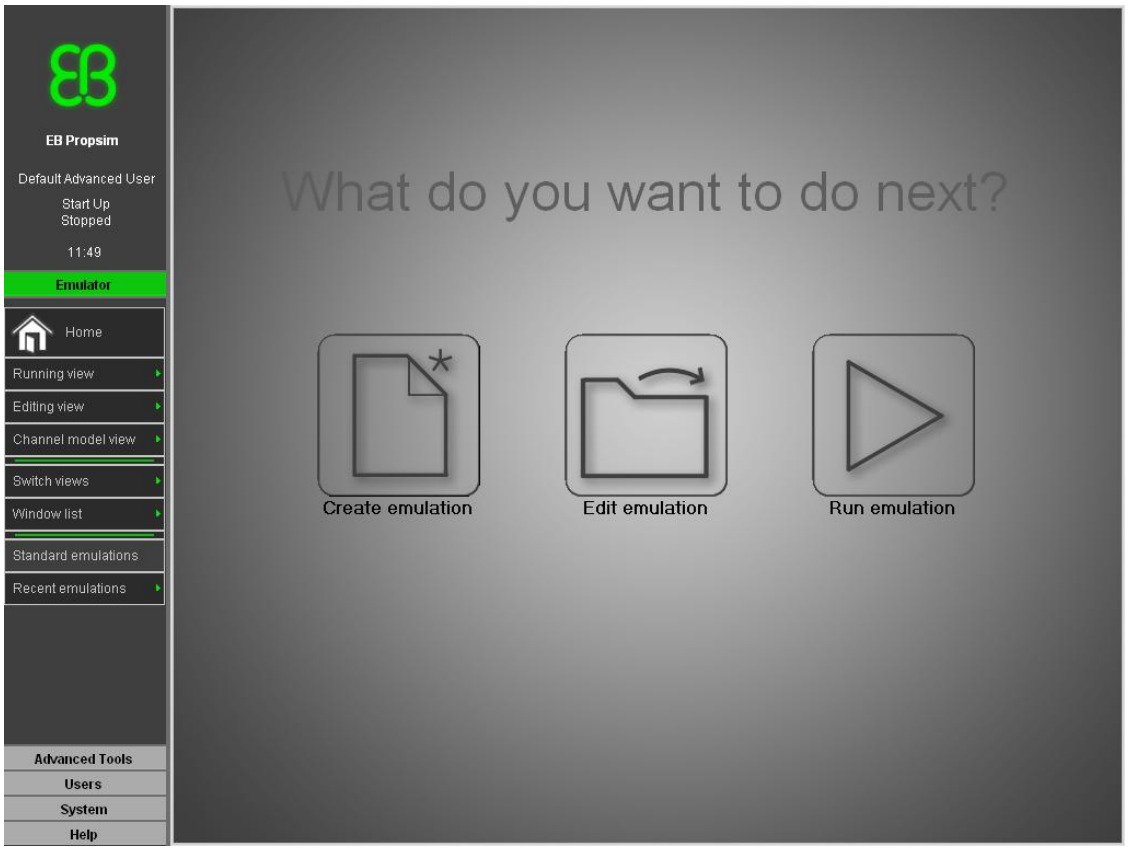
Työn tilaaja on Elektrobit System Test Oy, joka on Elektrobit Oyj:n tytäryhtiö. Elektrobit System Test Oy valmistaa radiokanavaemulaattoreita. Sen asiakkaita ovat mm. matkapuhelinvalmistajat, verkko-operaattorit ja siruvalmistajat.

2 EB PROPSIM -RADIOKANAVAEMULAATTORIT

EB Propsim -tuoteperheeseen kuuluu kolme radiokanavaemulaattoria: F8-, FS8- ja F32-radiokanavaemulaattorit, joista uusin on syyskuussa 2012 julkaistu FS8. Radiokanavaemulaattoreilla on mahdollista toistaa tarkasti langattomia ympäristöjä tuotteen tai järjestelmän suorituskyvyn testausta varten. Näin voidaan varmistaa tuotteen tai järjestemän toimivuus millä tahansa langattomalla teknologialla. (1.)

EB Propsim -radiokanavaemulaattorit emuloivat radiokanavien ominaisuuksia kuten etenemisvaimennusta, monitie-etenemistä, viivejakaumaa, Doppler-hajetta, polarisaatiota, korrelaatiota ja MIMO-järjestelmän toimivuudelle tärkeitä tilaparametrejä. (1.)

EB Propsim -radiokanavaemulaattoreiden käyttöliittymä (kuva 1) toimii Microsoft Windows -ympäristössä. Käyttöliittymän vasemmassa laidassa on navigointialue, josta voidaan vaihtaa eri näkymien välillä. Navigointialueella on mahdollista myös esimerkiksi luoda uusia käyttäjiä, poistaa käyttäjiä ja avata Windows-ohjelmia, kuten muistio ja resurssienhallinta.



KUVA 1. EB Propsim -radiokanavaemulaattoreiden käyttöliittymän päänäkymä

2.1 Emulaattorimallit

2.1.1 EB Propsim F8

EB Propsim F8 (kuva 2) on suunniteltu maanpäällisten, satelliitti- ja taktisten järjestelmien ja laitteiden testaamiseen. Sillä on mahdollista testata kaikkia olemassa olevia LTE-, LTE-Advanced-, WCDMA-, HSPA+-, GSM-, TD-SCDMA-, EV-DO- / CDMA2000-, TETRA-, IS-54-, WiMAX- ja Wi-Fi-radiojärjestelmiä. EB Propsim F8:n varusteluun kuuluu enintään 8 RF-rajapinnan kanavaa ja 64 häipymäkanavaa. (1.)

KUVA 2. EB Propsim F8 (2.)

2.1.2 EB Propsim F32

EB Propsim F32 (kuva 3) julkaistiin huhtikuussa 2012. Se on suunniteltu laaja-kaistaisten matkapuhelinverkkojen, kuten 2G/3G-, 3GPP LTE- ja LTE-Advanced-teknologioiden testaamiseen. EB Propsim F32:n varusteluun kuuluu enintään 32 RF-rajapinnan kanavaa ja 128 häipymäkanavaa. (1.)

KUVA 3. EB Propsim F32 (2.)

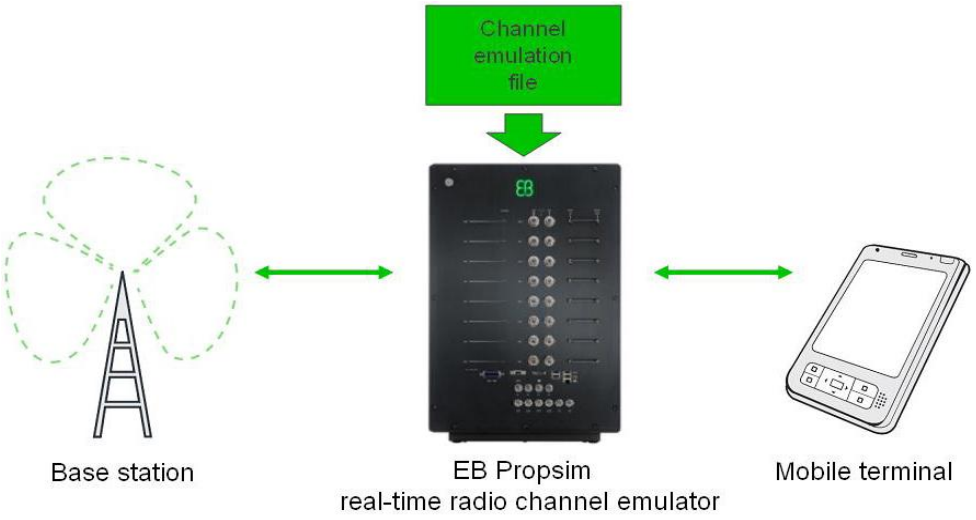
2.1.3 EB Propsim FS8

EB Propsim FS8 (kuva 4) on Elektrobitin uusin lisäys EB Propsim -radiokanavaemulaattoreiden tuoteperheeseen. FS8 on suunniteltu erityisesti LTE-verkon testaukseen. Se tukee kaistanleveyttä 40 MHz:iin ja kantotaajuutta 2700 MHz:iin asti. FS8 on varustettu 8 RF-rajapinnan kanavalla. (1.)

KUVA 4. EB Propsim FS8 (2.)

2.2 Emulaatiot ja niiden kääntäminen

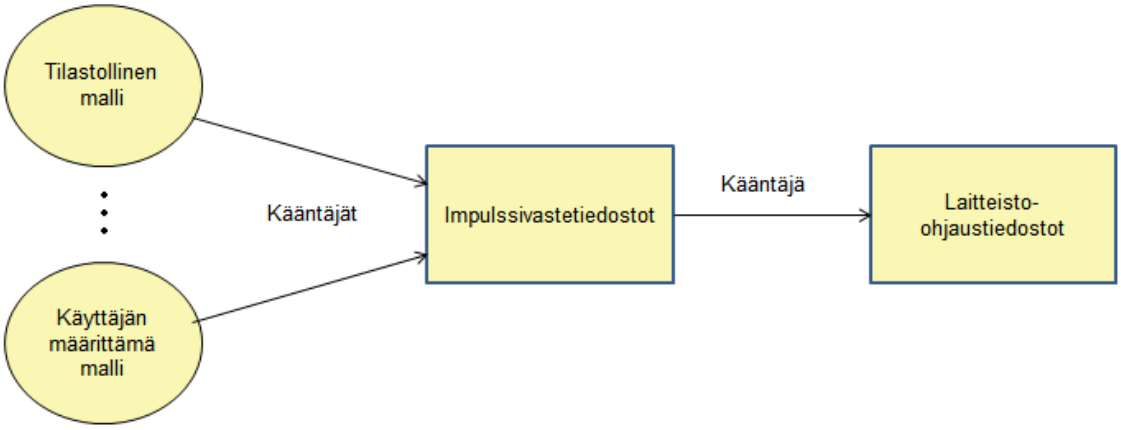
Emulaatiossa määritellään radioaaltojen etenemisympäristö järjestelmässä olevien eri langattoman tiedonsiirron laitteiden välille (kuva 5). Esimerkiksi tukiaseman ja kännykän välillä emulaatiossa määritellään ilmarajapinnan parametrit ja niiden muuttuminen emulaation ajon aikana.



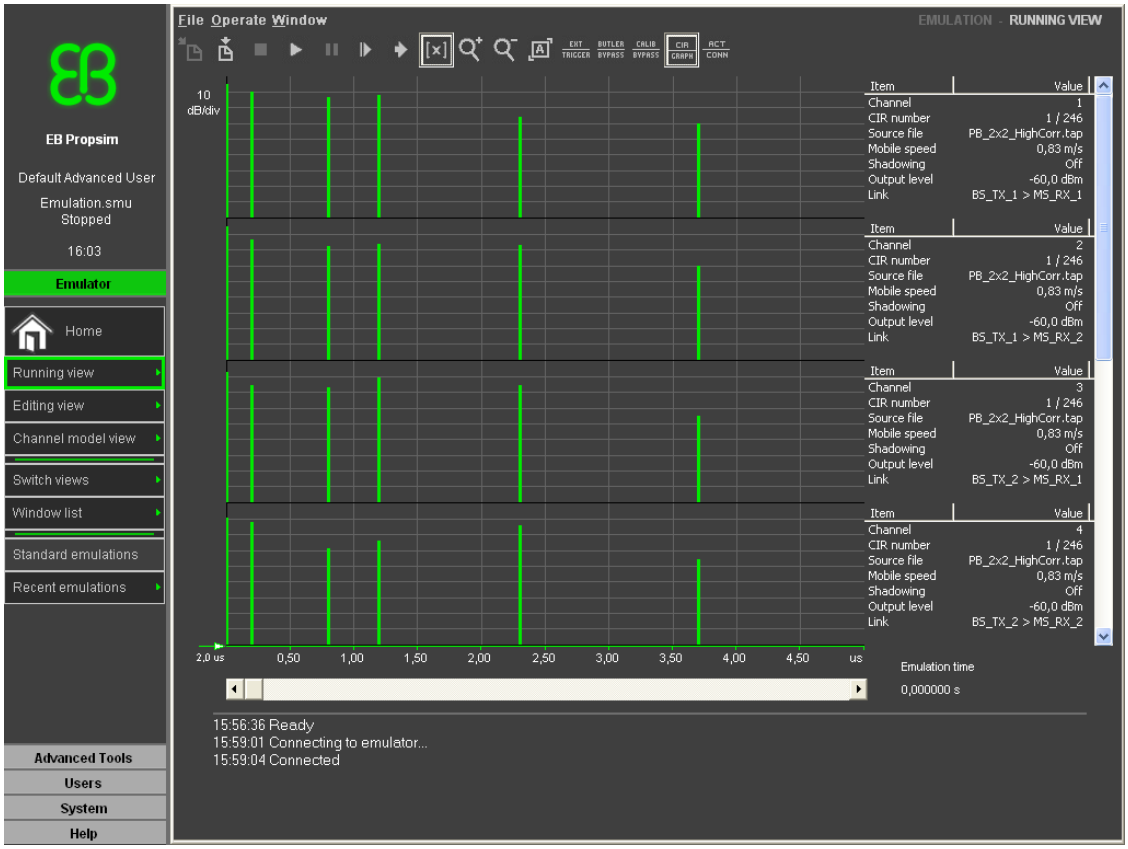
KUVA 5. Ilmarajapinnan korvaaminen radiokanavaemulaattorilla (3, s. 17)

Emulaation määritystiedosto sisältää mm. fyysisten liityntärajapintojen asetukset (esimerkiksi tehotasot, taajuudet jne.), emulaation ajoon liittyvät parametrit (esimerkiksi mobiilin liikkumisnopeus jne.) ja emulaation kääntämisessä tarvittavat tiedot, kuten tiedot kanavamalleista.

Emulaation kääntämisessä tilastolliset tai käyttäjän määrittelemät ilmarajapintaa kuvaavat mallit käännetään ensin impulssivastetiedostoiksi (.ir-tiedosto), jonka jälkeen ne käännetään laitteisto-ohjaustiedostoiksi (.sim-tiedosto) (kuva 6). Impulssivastetiedosto sisältää hetkelliset polkujen viive-, vaihe- ja amplitudi-informaatiot (kuva 7).



KUVA 6. Emulaation käännösprosessi



KUVA 7. 2x2 MIMO -emulaatio kanavan impulssivastenäkymässä

3 KÄYTETYT MENETELMÄT JA TYÖKALUT

3.1 .NET Framework

.NET Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto, jonka ensimmäinen versio julkaistiin tammikuussa 2002. Se sisältää useita luokkakirjastoja esimerkiksi Windows-, Windows Mobile- ja Web-ohjelmistojen kehitystä varten. Ohjelmistokehitys tapahtuu pääasiassa Microsoft Visual Studio -kehitysympäristössä. (4.)

3.2 C#

C# on Microsoftin vuonna 2000 julkaisema oliopohjainen ohjelmointikieli .NET-konseptia varten. Sen syntaksi muistuttaa C-kielen syntaksia. C#:ssa yhdistyy C++-kielen tehokkuus ja Java-kielen helppokäyttöisyys. Ensimmäinen ISO-standardoitu versio C#:sta julkaistiin vuoden 2002 tammikuussa. (5.)

3.3 C++/CLI

C++/CLI on Microsoftin kehittämä ohjelmointikieli, jolla on mahdollista käyttää .NET-alustan ominaisuuksia. C++/CLI-kielen syntaksi on Managed C++-kielen syntaksia yksinkertaisempi. (6.)

3.4 Microsoft Visual Studio 2005

Microsoft Visual Studio 2005 on Microsoftin kehittämä ohjelmistonkehitysympäristö. Sitä käytetään sekä konsoli- että graafisen käyttöliittymän omaavien ohjelmien, kuten Windows Forms -ohjelmien, tekemiseen. Se tukee mm. C/C++-, C#- ja Visual Basic -kieliä. (7.)

3.5 Windows Forms

Windows Forms (WinForms) on graafinen ohjelmointirajapinta, joka on osa Microsoft .NET Frameworkia. Se mahdollistaa alustariippumattoman tavan suunnitella graafisia käyttöliittymiä. Windows Forms on rakennettu Windows API:n päälle, joten sillä on mahdollista käyttää Microsoft Windows -rajapinnan elementtejä. (8.)

4 VAATIMUSMÄÄRITTELY

Vaatimusmäärittely suunniteltiin yhdessä tilaajan kanssa ennen työn aloittamista. Ohjelman käyttöliittymän suunnittelun malliksi saatiin tilaajan tekemä luonnos, josta saatiin hyvin selville, miltä käyttöliittymän pitää näyttää.

Ohjelman tulee avautua näytön alareunaan, kun se saa käskyn aloittaa emulaation kääntämisen. Ohjelmasta tulee pystyä näkemään emulaation käännösprosessin eteneminen, käännösprosessiin liittyviä tilaviestejä, seuraavana käännettäväksi tulevan emulaation nimi ja jonossa olevien emulaatioiden määrä. Jonossa olevien emulaatioiden määrä näkyy vain silloin, jos emulaatioita on jonossa enemmän kuin kaksi. Tilaviestiä klikkaamalla tulee avautua näkymä, josta näkee kaikki emulaation kääntämiseen liittyvät tilaviestit.

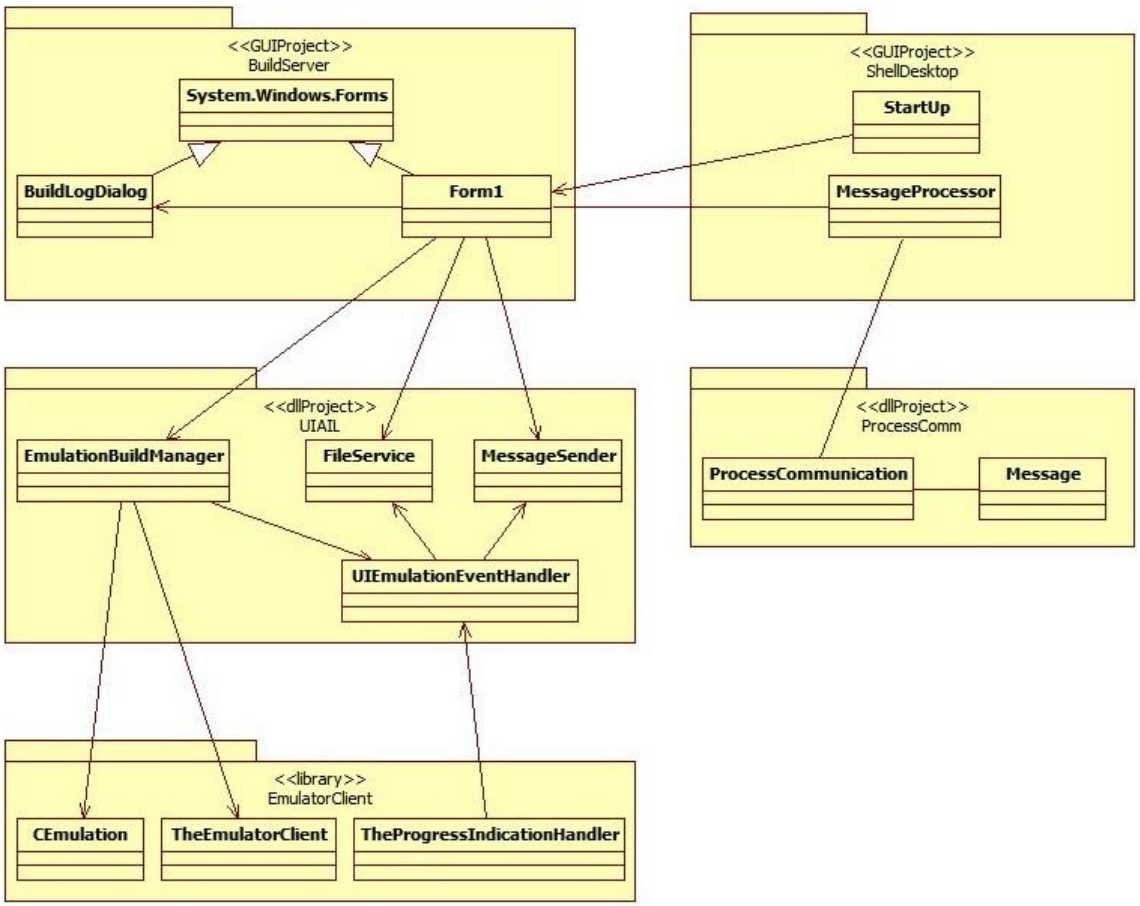
Emulaation kääntämisen aikana tulevat käännöspyynnöt menevät jonoon, josta seuraavana käännettäväksi tulee isoimmalla prioriteetilla oleva emulaatio tai ensimmäisenä jonoon lisätty emulaatio. Jos jonoon lisättävällä emulaatiolla on suurempi prioriteetti kuin kääntymässä olevalla emulaatiolla, ohjelman tulee kysyä, haluaako käyttäjä keskeyttää nykyisen käännöksen ja aloittaa uuden emulaation kääntämisen. Prioriteetin lisäksi jonossa oleville emulaatioille voidaan määrittää, avataanko emulaatio onnistuneen käännöksen jälkeen automaattisesti Running View'hun. Käyttäjän pitää myös pystyä avaamaan emulaatio manuaalisesti onnistuneen käännöksen jälkeen Running View'hun.

Käyttäjän tulee pystyä keskeyttämään emulaation kääntäminen, jolloin seuraavana jonossa oleva emulaatio siirtyy käännettäväksi. Jos jono on tyhjä, ohjelma piilotetaan taustalle tietyn ajan kuluttua.

5 OHJELMAN TOTEUTUS

Ohjelma toteutettiin Microsoft Visual Studio 2005 -kehitysympäristössä C#- ja C++/CLI-kieliä käyttäen. Ohjelmistokomponenttikirjastona käytettiin .NET Frameworkin versiota 2.0. Käyttöliittymän tekoon käytettiin Windows Forms -rajapintaa, joka on osa .NET Frameworkia. Käyttöliittymän ja C++:lla toteutetun kirjaston välille toteutettiin rajapinta C++/CLI-kielellä. Projektinhallintaan käytettiin JIRA-tehtävienhallintaohjelmistoa ja versionhallintaan StarTeamia.

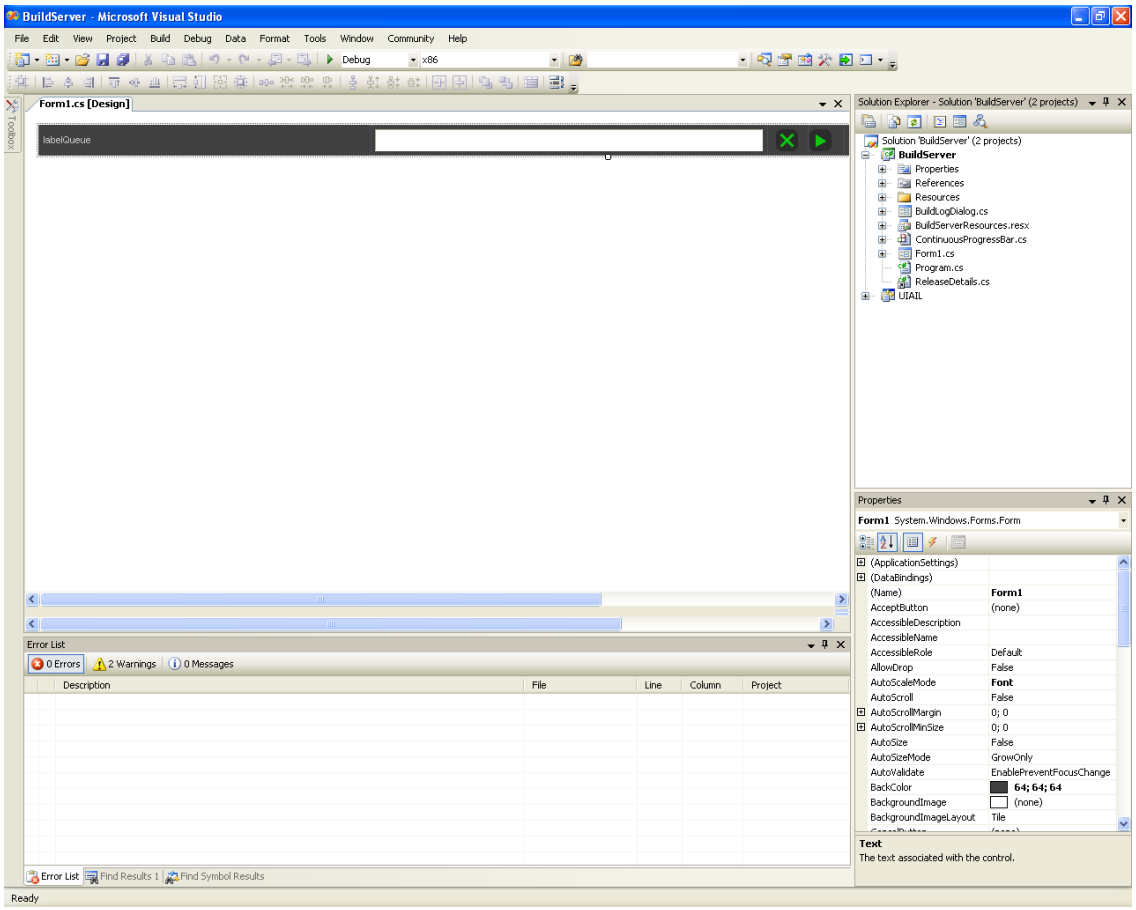
Ohjelmaan toteutettiin kaksi lomaketta, Form1 ja BuildLogDialog, joista Form1 toimii päälomakkeena. Ohjelman luokkakaaviosta on nähtävissä ohjelman rakenne ja luokkien väliset yhteydet (kuva 8).



KUVA 8. Ohjelman luokkakaavio

5.1 Käyttöliittymä

Ohjelman toteutus aloitettiin käyttöliittymän tekemisellä (kuva 9). Käyttöliittymästä pyrittiin tekemään mahdollisimman hyvin kaikilla näytön resoluutioilla toimiva.



KUVA 9. BuildServerin käyttöliittymä Microsoft Visual Studio 2005 -kehitysympäristössä

Käyttöliittymä toteutettiin TableLayoutPanel-kontrollin pohjalle, koska sillä on erittäin helppo luoda näytön resoluutiosta riippumattomia käyttöliittymiä. TableLayoutPanel-kontrolli koostuu soluista, joista jokaiseen voidaan asettaa vain yksi kontrolli. TableLayoutPanel-kontrollista luotiin periytetty kontrolli, jotta siihen saataisiin double buffering eli kaksoispuskurointi, joka vähentää kontrollin piirron välkkymistä.

Ensimmäiseen TableLayoutPanel-kontrollin soluun sijoitettiin Label-kontrolli, joka näyttää tällä hetkellä käännettävä olevan emulaation nimen, seuraavana jonossa olevan emulaation nimen ja jonossa olevien emulaatioiden lukumäärän. Toiseen soluun sijoitettiin ProgressBar-kontrolli, joka näyttää emulaation kääntämisen edistymisen. Kolmanteen ja neljänteen soluun sijoitettiin Button-kontrollit emulaation kääntämisen keskeyttämistä ja emulaation Running View'hun avaamista varten. Viimeiseen soluun sijoitettiin Label-kontrolli, joka näyttää emulaation kääntämiseen liittyviä tilaviestejä.

5.2 Emulaation kääntäminen

Emulaation kääntämiseen liittyvän toteutuksen tekeminen aloitettiin luomalla UIAIL-projektiin (UI Access Interface Layer) EmulationBuildManager-luokka BuildServerin ja kirjaston välistä viestintää varten. UIAIL on C++/CLI-kielellä kirjoitettu rajapinta C#:lla toteuttujen ohjelmien ja C++-kielellä toteutettujen kirjastojen välillä.

EmulationBuildManager-luokkaan toteutettiin public-tyyppinen BuildEmulation-funktio, joka ottaa vastaan emulaation määritystiedoston polun string-tyyppinä.

BuildEmulation-funktiossa emulaatio avataan kirjaston CEmulation-luokan openEmulationFile-funktiota käyttäen, joka tarkistaa myös käytössä olevat lisenssit. Emulaation kääntäminen aloitetaan, kun emulaatio on avattu onnistuneesti. Emulaation kääntämisen aloitus tapahtuu kutsumalla IEmulation-luokan build-funktiota, joka palauttaa BuildStartStatus-tyyppisen string-viestin emulaation kääntämisen aloituksen tilasta. BuildStartStatus-viestin tarkistus toteutettiin if-lausekkeilla. Jos käännöksen aloituksessa tapahtuu virhe, käyttöliittymälle palautetaan virheviesti rajapinnan MessageSender-luokan SendMessage-funktiota käyttäen ja emulaation kääntäminen keskeytetään.

Onnistuneen kääntämisen aloituksen jälkeen kutsutaan InitializeTimer-funktiota, jossa alustetaan ja käynnistetään Timer-kontrolli tilaviestien käsittelyä varten. Timer-kontrollin Tick-tapahtumankäsittelijään toteutettiin kirjastolta tulevien emulaation kääntämiseen liittyvien tilaviestien käsittely ja muuttaminen yhteen-

sopiviksi C#:n kanssa. Tilaviestit saadaan kirjastolta kutsumalla CEmulation-luokan getBuildOutput-metodia, joka palauttaa tilaviestit char-tyyppisenä merkkijonona. Tilaviestit käsitellään for-silmukassa, jossa merkkijonoa luetaan merkki kerrallaan niin kauan, kunnes vastaanotetaan rivinvaihtomerkki. Tämän jälkeen merkkijono luetaan .NET-yhteensopivaan string-muuttujaan ja lähetetään eteenpäin rajapinnan MessageSender-luokan SendMessage-funktiolla, jossa viesti lähetetään StringMessage-tapahtumaan.

Kun emulaation kääntäminen on valmis, kutsutaan CEmulation-luokan areEmulationFilesUpToDate-funktiota, joka palauttaa true-arvon, jos emulaation kääntäminen on onnistunut. Tämän jälkeen BuildServerille lähetetään viesti emulaation käännösprosessin tuloksesta ja suljetaan emulaatio sekä pysäytetään Timer-kontrolli.

Emulaation kääntämisen keskeyttämistä varten luotiin Cancel-napin klikkaukselle tapahtumankäsittelijä, jossa kutsutaan cancelBuilding-funktiota, jos emulaation kääntäminen on käynnissä. CancelBuilding-funktioon toteutettiin UIAIL-rajapinnan EmulationBuildManager-luokassa olevan stopBuild-funktion kutsuminen ja emulaation kääntämiseen liittyvien prosessien sulkeminen. StopBuild-funktiossa kutsutaan kirjaston CEmulation-luokan StopBuild-funktiota, joka keskeyttää emulaation kääntämisen.

Emulaation avaaminen Running View’hun toteutettiin runEmulation-funktioon, jossa lähetetään viesti ShellDesktopille ProcessComm-projektin ProcessCommunication-luokassa olevaa SendMessageToProcess-funktiota käyttäen. RunEmulation-funktiota kutsutaan, kun emulaation kääntäminen on suoritettu onnistuneesti ja emulaation avaaminen on ollut valittuna. Käyttäjän on mahdollista aktivoida emulaation avaaminen Running View’hun joko Scenario Wizardin Build&Run-nappia painamalla tai klikkaamalla Run-nappia emulaation kääntämisen aikana, jolloin Run-napin taustaväri muuttuu harmaasta vihreäksi. Run-toiminnon tarkistus tehdään, kun emulaatio siirtyy käännettäväksi. Se tapahtuu tarkistamalla Emulation-struktuurin Run-parametrin arvo. Jos emulaatio on avattava Running View’hun, asetetaan runAfterBuildingDone-booleanin arvoksi true.

RunAfterBuildingDone-booleanin tarkistus tapahtuu, kun emulaatio on käännetty onnistuneesti. Jos arvo on true, kutsutaan runEmulation-funktiota.

5.2.1 Käännöspyyntöjen käsittely

Emulaation käännöspyynnön lähetys Scenario Wizardilta BuildServerille tapahtuu siten, että Scenario Wizard lähettää käännöspyynnön ShellDesktopille, joka välittää viestin ProcessComm-projektin kautta BuildServerille. Viestin lähetys toteutettiin ProcessComm-projektiin, jonka ProcessCommunication-luokkaan toteutettiin SendBuildMessage-funktio. Funktio ottaa vastaan PCMessage-luokan olion, johon toteutettiin uusi konstruktori käännöspyyntöviestin rakentamista varten. Konstruktori ottaa vastaan viestityypin, viestin id:n ja BuildMsg-struktuurin. BuildMsg-struktuuri sisältää emulaation polun, ohjelman nimen, josta käännöspyyntö lähetettiin, prioriteetin ja tiedon emulaation avaamisesta Running View'hun.

SendBuildMessage-funktioon toteutettiin käännösviestin lähetys BuildServerille. Funktiossa tarkistetaan, onko BuildServer-prosessi käynnissä, ja tarvittaessa käynnistetään BuildServer. Lähetettävä viesti konvertoidaan IntPtr-tyyppiseksi, jotta se voidaan lähettää BuildServerille. Viesti lähetetään Windows APIn SendMessage-funktiota käyttäen, joka tarvitsee viestin vastaanottavan ohjelman MainWindowHandlen, jotta viesti menee perille. BuildServerin MainWindowHandlen saamiseksi jouduttiin tekemään while-silmukka, joka kutsuu prosessin Refresh-metodia niin kauan, kuin MainWindowHandlen arvo on nolla. Tämä tehtiin sen takia, koska prosessilla ei ole MainWindowHandlea silloin, kun sillä ei ole näkyviä ikkunoita.

BuildServer kuuntelee viestejä WndProc-funktiossa. WndProc kuuluu Windows APIin ja se vastaanottaa kaikki ohjelmalle tulevat viestit. Kun BuildServer vastaanottaa käännöspyyntöviestin, se luo uuden BuildMsg-struktuurin ja lisää siihen vastaanotetut tiedot. Tämän jälkeen kutsutaan AddEmulationToQueue-funktiota, joka ottaa vastaan BuildMsg-struktuurin. AddEmulationToQueue-funktiossa emulaatio lisätään BuildingQueue.xml-tiedostoon.

BuildingQueue.xml-tiedosto sisältää käännettäväksi tulevien emulaatioiden tiedot DataTable-taulukossa, josta aina ensimmäisenä oleva emulaatio otetaan käännettäväksi. BuildingQueue.xml-tiedoston rakenne on nähtävissä kuvassa 10. Jos tiedosto löytyy, luodaan uusi DataSet, johon BuildingQueue.xml-tiedosto ladataan. Xml-tiedosto ladataan DataSetiin ReadXml-komennolla, jolle annetaan xml-tiedoston polku ja XmlReadMode, joka määrittää, kuinka xml-tiedosto luetaan.

```
<?xml version="1.0" standalone="true"?>
<queueDataSet xmlns="BuildingQueue">
  - <xs:schema xmlns="BuildingQueue" elementFormDefault="qualified" attributeFormDefault="qualified"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:mstns="BuildingQueue" targetNamespace="BuildingQueue" id="queueDataSet">
      - <xs:element msdata:UseCurrentLocale="true" msdata:IsDataSet="true" name="queueDataSet">
          - <xs:complexType>
              - <xs:choice maxOccurs="unbounded" minOccurs="0">
                  - <xs:element name="QueueTable">
                      - <xs:complexType>
                          - <xs:sequence>
                                <xs:element name="Emulation" minOccurs="0" type="xs:string"/>
                                <xs:element name="Priority" minOccurs="0" type="xs:int"/>
                                <xs:element name="Caller" minOccurs="0" type="xs:string"/>
                                <xs:element name="Run" minOccurs="0" type="xs:string"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
            </xs:complexType>
        </xs:element>
    </xs:schema>
  - <QueueTable>
        <Emulation>C:\EB Propsim F8\Simulated F8\My emulations\Emulation.wiz\Emulation.smu</Emulation>
        <Priority>1</Priority>
        <Caller>Scenario Wizard</Caller>
        <Run>0</Run>
    </QueueTable>
  - <QueueTable>
        <Emulation>C:\EB Propsim F8\Simulated F8\My emulations\Emulation2.wiz\Emulation2.smu</Emulation>
        <Priority>1</Priority>
        <Caller>Scenario Wizard</Caller>
        <Run>0</Run>
    </QueueTable>
  - <QueueTable>
        <Emulation>C:\EB Propsim F8\Simulated F8\My emulations\Emulation3.wiz\Emulation3.smu</Emulation>
        <Priority>1</Priority>
        <Caller>Scenario Wizard</Caller>
        <Run>0</Run>
    </QueueTable>
  - <QueueTable>
        <Emulation>C:\EB Propsim F8\Simulated F8\My emulations\Emulation4.wiz\Emulation4.smu</Emulation>
        <Priority>1</Priority>
        <Caller>Scenario Wizard</Caller>
        <Run>0</Run>
    </QueueTable>
</queueDataSet>
```

KUVA 10. BuildingQueue.xml-tiedoston rakenne

Emulaation sijoittaminen oikealle paikalle käännösjonossa toteutettiin if-lausekkeilla, joissa vertaillaan emulaatioiden prioriteettia. Jos lisättävänä olevan emulaation prioriteetti on suurempi kuin kääntymässä olevan emulaation, ohjelma kysyy käyttäjältä, keskeytetäänkö nykyinen käännös ja aloitetaan lisättävän emulaation kääntäminen. Funktion lopussa aloitetaan emulaation kääntäminen aktivoimalla emulationBuildThread-säie, jos emulaation kääntäminen ei ole jo käynnissä. EmulationBuildThread-säikeessä kirjoitetaan BuildLog.log-tiedostoon käännettävänä olevan emulaation polku ja aloitetaan emulaation kääntäminen kutsumalla rajapinnan EmulationBuildManager-luokan BuildEmulation-funktiota. BuildLog.log-tiedostoon tallennetaan kaikki emulaation kääntämisen aikana tulevat tilaviestit. BuildLog.log-tiedoston rakenne on nähtävissä kuvassa 11.

```
[BuildStart]
<Lime>15:12:44 Building: C:\EB Propsim F8\Simulated F8\My emulations\Emulation.wiz\Emulation.smu

15:12:47 TapToIrCompiler.exe -s  --split 2 -i "GSM_EqualisationTest_6Taps.tap" -o "GSM_EqualisationTest_6Taps_0"
15:12:49 Generating IR-file(s)...

15:13:32 IrToSimCompiler.exe -hw "C:\Priki\Debug\default.hw" --split 2 --timesplit 4 -hwi -b 64 -ee2 -i
"GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_0_S2_T4.sim"
15:13:32 Collecting statistics from IR-file...
15:13:48 Allocating paths to hardware...
15:14:06 Fading block count=2
15:14:06 Generating SIM-file...
15:14:29 Channel model used 1 fading channel unit.

15:14:29 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_1.ir"
15:14:30 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_1_S2_T4.sim"
15:14:30 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_2.ir"
15:14:30 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_2_S2_T4.sim"
15:14:31 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_3.ir"
15:14:31 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_3_S2_T4.sim"
15:14:32 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_4.ir"
15:14:33 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_4_S2_T4.sim"
15:14:33 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_5.ir"
15:14:33 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_5_S2_T4.sim"
15:14:33 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_6.ir"
15:14:34 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_6_S2_T4.sim"
15:14:34 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0.ir" -o "GSM_EqualisationTest_6Taps_7.ir"
15:14:34 FileCopy.exe -t -i "GSM_EqualisationTest_6Taps_0_S2_T4.sim" -o "GSM_EqualisationTest_6Taps_7_S2_T4.sim"
15:14:35 SIM-file generation done

<Lime>15:14:35 Emulation built succesfully.
[BuildEnd]
```

KUVA 11. BuildLog.log-tiedoston rakenne

Kun emulaation kääntäminen on suoritettu, kutsutaan UpdateBuildingQueue-funktiota. Tähän funktioon toteutettiin BuildingQueue.xml-tiedoston tarkistus. Jos jonossa on emulaatioita, aktivoidaan emulationBuildThread-säie ja päivitetään käyttöliittymässä olevat Label-kontrollit. Käännösjonon ollessa tyhjä käynnistetään Timer-kontrolli, joka piilottaa BuildServerin tietyn ajan kuluttua.

5.2.2 Tilaviestien ja kääntämisen edistymisviestien käsittely

Tilaviestit ja kääntämisen edistymisviestit saatiin rajapinnalta käyttöliittymälle luomalla päälomakkeeseen tapahtumankäsittelijät rajapinnassa olevan MessageSender-luokan StringMessage- ja ProgressMessage-tapahtumiin.

Tilaviestit tulevat ohjelmalle päälomakkeeseen toteutetun StringMessage-tapahtumankäsittelijän kautta. Tapahtumankäsittelijässä viestit käsitellään if-lausekkeissa ja lisätään rivi kerrallaan BuildLog.log-tiedostoon. StringMessage-tapahtumankäsittelijässä asetetaan myös väritageja esimerkiksi virheviestien eteen. Esimerkiksi asettamalla <Red>-tagi virheviestin eteen kyseinen rivi tulostuu BuilLogDialogissa punaisella värillä.

Emulaation kääntämisen etenemiseen liittyvät viestit tulevat ohjelmalle päälomakkeeseen toteutetun ProgressMessage-tapahtumankäsittelijän kautta.

Tapahtumankäsittelijässä asetetaan vastaanotettu luku ContinuousProgressControliin, joka on ProgressControl-luokasta periytetty kontrolli.

Tilaviestien tarkempaa tarkastelua varten luotiin BuildLogDialog-lomake. Lomake aukeaa, kun klikataan tilaviestien Label-kontrollia päälomakkeessa. BuildLog.log-tiedosto luetaan RichTextBox-kontrolliin loadBuildLog-funktiossa, kun lomake avataan. BuildLogDialogin ollessa auki BuildLog.log-tiedoston muutoksia tarkkaillaan FileSystemWatcherilla, joka lisää jokaisen tekstiedostoon tulevan uuden rivin BuildLogDialogissa olevaan RichTextBox-kontrolliin. FileSystemWatcher luodaan BuildLogDialogin konstruktorissa, jossa sille luodaan Changed-tapahtumankäsittelijä. Changed-tapahtumankäsittelijää kutsutaan aina, kun FileSystemWatcher havaitsee muutoksia BuildLog.log-tiedostossa. Changed-tapahtumankäsittelijässä kutsutaan readLastLine-funktiota, jossa uudet rivit lisätään RichTextBox-kontrolliin.

ReadLastLine-funktiossa käydään BuildLog.log-tiedosto läpi siitä rivistä alkaen, mihin loadBuildLog-funktiossa jäätiin, kun BuildLogDialog avattiin, tai siitä, mihin readLastLine-funktiossa viimeksi jäätiin.

Ohjelman käynnistyksen yhteydessä kutsutaan päälomakkeessa olevaa parseBuildLog-funktiota, johon toteutettiin BuildLog.log-tiedoston tarkistus. Funktiossa tarkistetaan tiedostossa olevien emulaatioiden määrä laskemalla emulaatioiden määrä lokitiedostossa olevien [BuildStart]-tagien avulla, jotka lisätään lokitiedostoon emulaation kääntämisen aloituksen yhteydessä. Jos emulaatioita on yli viisi, poistetaan lokitiedostosta emulaatioiden tilaviestit vanhimmasta emulaatiosta alkaen.

5.3 Integrointi EB Propsim -ohjelmistoon

BuildServer integroitiin ShellDesktopiin siten, että se käynnistyy taustalle piilotettuna aina ShellDesktopin käynnistyksen yhteydessä. Alun perin BuildServerin piti käynnistyä vain silloin, kun se vastaanottaa käännöspyynnön. Tästä ajatuksesta kuitenkin luovuttiin, koska ohjelman käynnistykseen saattoi kulua jopa 5–10 sekuntia.

BuildServerin koko määräytyy ShellDesktopin mukaan. Koon ja sijainnin asettaminen toteutettiin päälomakkeeseen setFormLocation-funktioon, jossa tarkistetaan onko ShellDesktop käynnissä. Jos ShellDesktop on käynnissä, BuildServer hakee ShellDesktopin sijainnin ja sijoittuu sen oikealle puolelle. ShellDesktopin sijainti ja koko saadaan kutsumalla Windows APIn GetWindowRect-funktiota.

5.4 Testaus

Ohjelmaa testattiin koko kehitystyön ajan työpisteellä ja myöhemmässä vaiheessa myös laboratoriossa. Testausalustana laboratoriossa käytettiin eri EB Propsim -radiokanavaemulaattoreita. Testausta laitteessa alettiin tehdä heti, kun BuildServer saatiin integroitua ShellDesktopiin ja kääntämään emulaatio onnistuneesti. Työpisteellä ohjelmaa testattiin aina, kun siihen oli tehty jokin uusi ominaisuus, jotta mahdolliset virheet löytyisivät nopeasti.

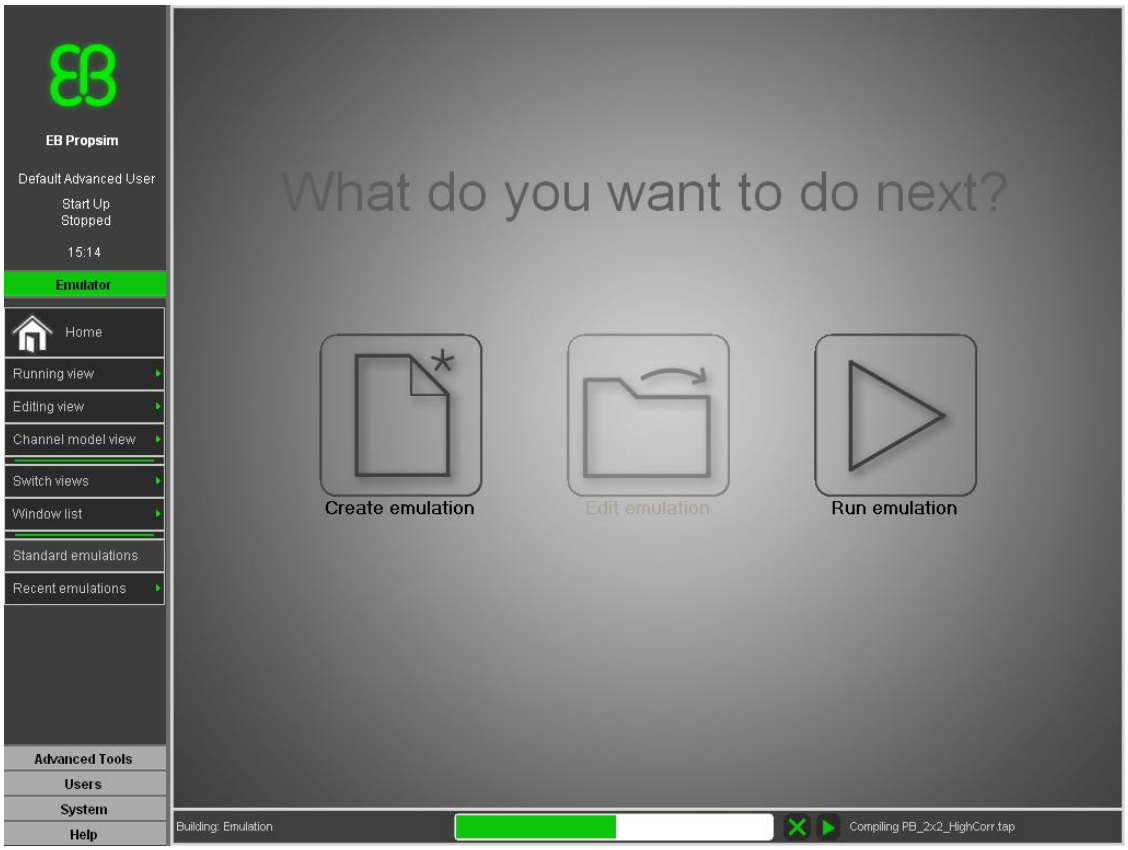
Testauksen aikana huomattiin useita ongelmia. Esimerkiksi ohjelman avaamisessa oli viivettä hitaammilla prosessoreilla varustetuilla radiokanavaemulaatto-

reilla, kun ohjelma ei ollut taustalla piilotettuna. Ohjelman ajokansion hakemisessa huomattiin myös ongelmia Application.StartupPath-toimintoa käytettäessä. Tämän takia ajokansion hakemisessa jouduttiin siirtymään kirjaston käyttämiseen. Testauksen aikana löydetyt ohjelmavirheet kirjattiin JIRAan.

6 VALMIIN OHJELMAN ESITTELY

Ohjelma käynnistyy taustalle piilotettuna EB Propsim -radiokanavaemulaattorin ohjelmiston käynnistyksen yhteydessä. Ohjelma liukuu näkyville näytön alareunasta, kun se saa Scenario Wizardilta käännöspyynnön. Emulaation kääntäminen aloitetaan automaattisesti, kun ohjelma on avautunut.

Emulaation kääntämisen aikana käyttäjä näkee ohjelmasta kääntymässä olevan emulaation nimen, kääntämisen etenemisen ja siihen liittyviä tilaviestejä, joista tärkeimmät näytetään päänäkymässä (kuva 12).



KUVA 12. Emulaation kääntäminen käynnissä

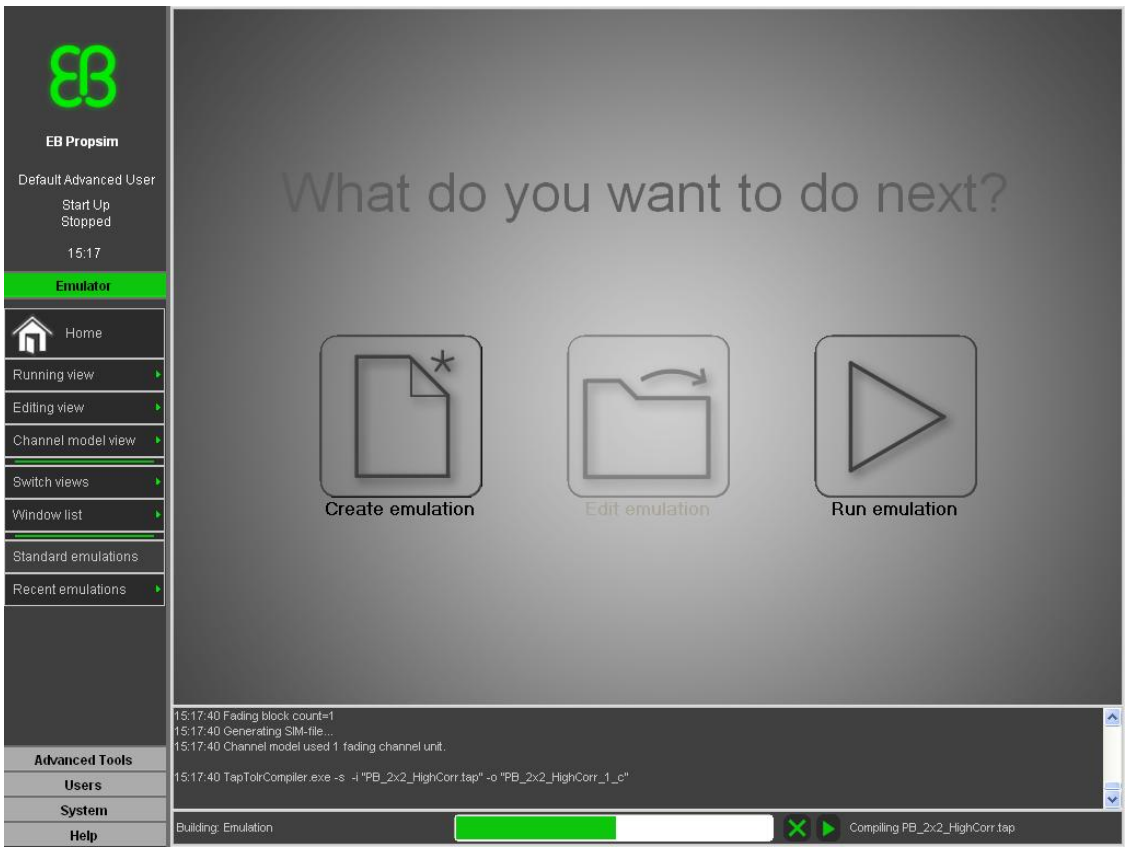
Emulaation kääntämisen aikana ohjelman vasemmassa laidassa näkyy käännösjonoon liittyviä tietoja. Jos käännösjonossa on kaksi emulaatiota, näytetään ainoastaan seuraavana käännettäväksi tulevan emulaation nimi (kuva 13).

Jos käännösjonossa on enemmän kuin kaksi emulaatiota, näytetään myös jonossa olevien emulaatioiden määrä (kuva 14).

KUVA 13. Seuraavana jonossa olevan emulaation nimi

KUVA 14. Jonossa olevien emulaatioiden määrä

Ohjelman oikeassa laidassa olevaa aluetta klikkaamalla on mahdollista avata käännöslokinäkymä, josta näkee kaikki emulaation kääntämiseen liittyvät tilaviestit sekä viiden aikaisemmin käännetyn emulaation tilaviestit (kuva 15).



KUVA 15. Käännöslokinäkymä avoinna

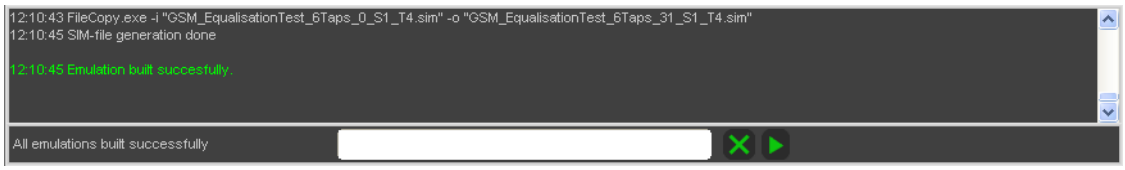
Emulaation kääntämisen aikana tapahtuvista virheistä ilmoitetaan erilaisilla virheteksteillä. Virheilmoitus näytetään sekä käännöslokinäkymässä että päänäkymässä (kuva 16 ja 17). Virheilmoituksen jälkeen ohjelma tarkistaa käännösjonon ja ohjelma suljetaan, jos jonossa ei ole kääntämättömiä emulaatioita.

KUVA 16. Virheilmoitus käännöslokinäkymässä

KUVA 17. Virheilmoitus päänäkymässä ja käännöslokinäkymässä

Kun emulaatio on käännetty onnistuneesti, ohjelma ilmoittaa siitä ”Emulation built succesfully” -tekstillä oikeassa laidassa olevalla tilaviesteillä tarkoitetulla alueella (kuva 18). Sama ilmoitusteksti näkyy myös käännöslokinäkymässä vihreällä värillä. Tämän jälkeen ohjelma tarkistaa käännösjonon. Jos jono on tyhjä ja kaikki emulaatiot on käännetty onnistuneesti, ohjelma ilmoittaa siitä ”All emulations built successfully” -tekstillä ennen sulkeutumista (kuva 19).

KUVA 18. Emulaatio käännetty onnistuneesti

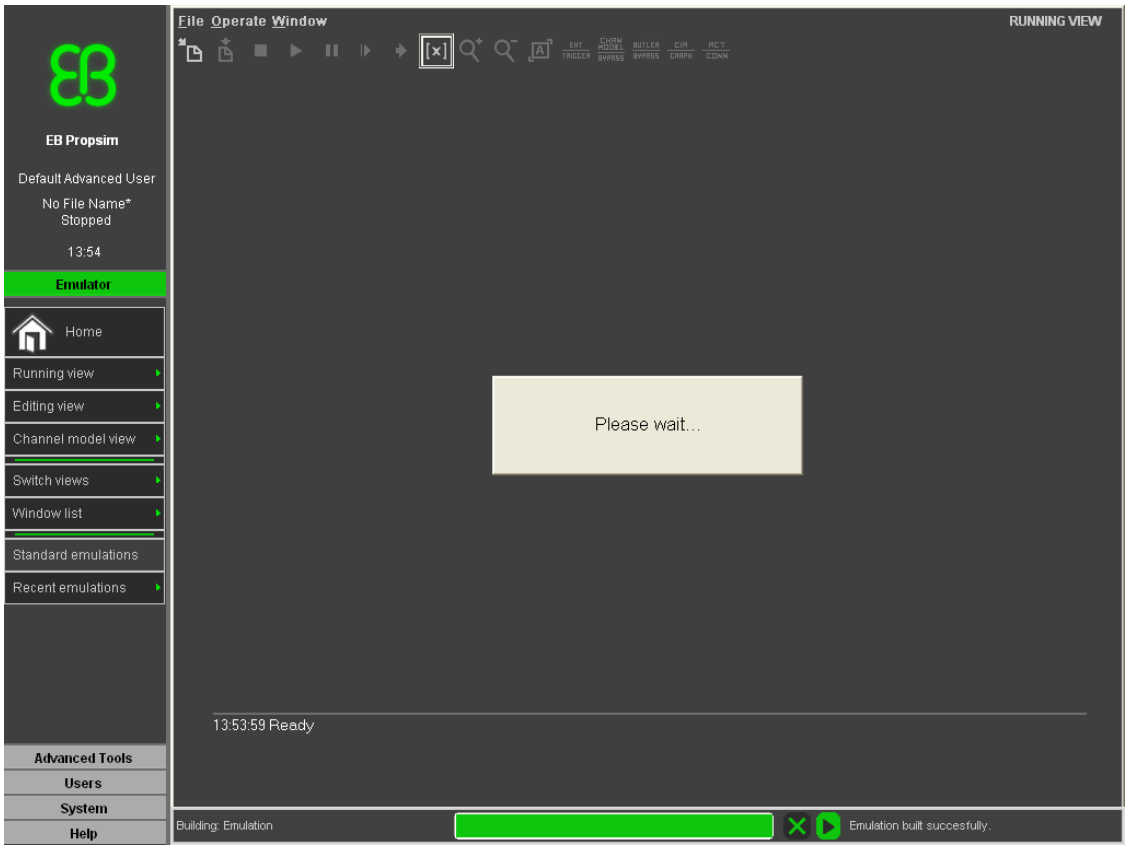


KUVA 19. Kaikki jonossa olleet emulaatiot käännetty onnistuneesti

Emulaation kääntäminen on mahdollista keskeyttää painamalla Cancel-nappia, jolloin ohjelma ilmoittaa ”Building canceled” -tekstillä kääntämisen keskeytyneen. ”Building canceled” -teksti näkyy myös käännöslokinäkymässä punaisella värillä (kuva 20). Käännöksen keskeytymisen jälkeen ohjelma tarkistaa käännösjonon ja sulkeutuu, jos jono on tyhjä.

KUVA 20. Emulaation kääntäminen keskeytetty

Käännettävänä oleva emulaatio on mahdollista avata onnistuneen käännöksen jälkeen Running View -näkymään (kuva 21). Tämä tapahtuu klikkaamalla Cancel-napin vieressä olevaa Play-nappia. Napin taustaväri muuttuu vihreäksi, kun emulaation avaaminen on aktivoitu. Edellisen onnistuneesti käännetyn emulaation avaaminen on mahdollista myös silloin, kun seuraavana jonossa olevan emulaation kääntäminen on jo käynnissä.



KUVA 21. Emulaatio avautumassa onnistuneen käännöksen jälkeen Running View'hun

7 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa EB Propsim -radiokanavaemulaattoreiden uutta käyttöliittymää varten ohjelma emulaatioiden kääntämiseen. Ohjelman oli tarkoituksena toimia emulaatioiden käännöspalvelimena, joka käsittelee ja suorittaa muilta ohjelmilta tulevat käännöspyynnöt käyttäen olemassa olevia kääntäjiä ja näyttää käyttäjälle tietoja emulaation käännösprosessista. Ohjelman tuli myös mahdollistaa usean emulaation asettaminen käännösjonoon, jolloin ohjelma kääntää automaattisesti kaikki jonossa olevat emulaatiot. Työn tavoitteissa onnistuttiin hyvin ja tuloksena saatiin tilaajan asettamien vaatimusten mukainen ohjelma.

Ohjelma käsittelee tulevaisuudessa mahdollisesti myös muilta kuin Scenario Wizardilta tulevat käännöspyynnöt. Lisäksi ohjelmaan on tarkoitus lisätä uusia ominaisuuksia ja sen toimintaa on tarkoitus muuttaa siten, että se toimii samalla tavalla kuin Windowsin tehtäväpalkki, jonka automaattinen piilottaminen on otettu käyttöön.

Työ oli mielenkiintoinen, koska olin aikaisemminkin tehnyt EB Propsim -radiokanavaemulaattoreiden käyttöliittymään liittyvää suunnittelutyötä. Työ oli minulle erittäin hyödyllinen, sillä sain lisää kokemusta Windows Forms -ohjelmien tekemisestä ja C#- sekä C++/CLI-ohjelmointikielistä, joista on varmasti hyötyä tulevaisuudessa työelämässä.

LÄHTEET

1. EB Propsim Radio Channel Emulator. 2012. Saatavissa: http://www.elektrobit.com/index.php?3451. Hakupäivä 7.10.2012.

2. EB - Wireless Products. 2013. Saatavissa: http://www.elektrobit.com/media/picture_gallery/wireless_products. Hakupäivä 23.1.2013.

3. EB Propsim User Reference. 2012. Käyttöopas. Elektrobit System Test Oy.

4. .NET Framework Conceptual Overview. 2012. Saatavissa: http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.80).aspx. Hakupäivä 6.10.2012.

5. C Sharp (programming language). 2012. Saatavissa: http://en.wikipedia.org/wiki/C_Sharp_(programming_language). Hakupäivä 6.10.2012.

6. Pure C++: Hello, C++/CLI. 2006. Saatavissa: http://msdn.microsoft.com/en-us/magazine/cc163681.aspx. Hakupäivä 6.10.2012.

7. Microsoft Visual Studio 2005. 2012. Saatavissa: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2005. Hakupäivä 6.10.2012.

8. Windows Forms. 2012. Saatavissa: http://en.wikipedia.org/wiki/Windows_Forms. Hakupäivä 7.10.2012.

9. Windows Forms Reference. 2012. Saatavissa: http://msdn.microsoft.com/en-US/library/ms229608(v=vs.80).aspx. Hakupäivä 6.12.2012.