Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Pasi Parkkonen

Graafinen konfigurointityökalu

Insinöörityö 30.4.2009

Ohjaaja: yliopettaja Matti Puska
Ohjaava opettaja: yliopettaja Kari Aaltonen

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä	Pasi Parkkonen
Otsikko	Graafinen konfigurointityökalu
Sivumäärä	49 sivua
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaaja	yliopettaja Matti Puska
Ohjaava opettaja	yliopettaja Kari Aaltonen

Tässä insinöörityössä tehtiin graafinen konfigurointityökalu, jolla on mahdollista tehdä reitittimen asetukset. Työn tavoitteena oli suunnitella ja toteuttaa kyseinen ohjelma.

Ohjelman suunnittelussa on käytetty perustana nelikerrosarkkitehtuuria ja se on kehitetty kahdessa vaiheessa. Vaiheessa yksi on suoritettu prototyypin kehitys ja vaiheessa kaksi tuoteversio 1.0. Tuotteen versio 1.0:n arkkitehtuuri on suunniteltu koostumaan neljästä kerroksesta, joista käyttöliittymäkerros, sovelluslogiikkakerros ja tiedonsaantikerros ovat ohjelman sisäisiä kerroksia. Neljäs kerros eli säilytyskerros on ohjelman ulkopuolinen kerros.

Ohjelman toteutuksessa on käytetty C#-ohjelmointikieltä, ja se on ohjelmoitu Visual Studio 2008 -kehitystyökalulla. Ohjelman testaukset suoritettiin Visual Studio 2008 -kehitystyökalun testiprojektina.

Työn tuloksena on saatu toimiva ohjelma, vaikka kaikkia asiakasvaatimuksia ei ole täytetty.

Hakusanat	konfigurointityökalu, nelikerrosarkkitehtuuri, olioperustainen, NET-ohjelmistokehys

Helsinki Metropolia University of Applied Sciences **Abstract**

Author	Pasi Parkkonen
Title	Graphical configuration tool
Number of Pages	49 pages
Date	30 April 2009
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor	Matti Puska, Principal Lecturer
Supervisor	Kari Aaltonen, Principal Lecturer

The purpose of this engineering thesis was to create a Graphical Configuration Tool, which has a possibility to do configuration tasks for a router. The goal of this thesis was to design and implement this software.

The fourth layered architecture has been used as a baseline for the design and the thesis has been developed in two phases. In phase one a prototype has been developed and in phase two a product version 1.0 has been implemented. The architecture of product version 1.0 has been planned to consist of four layers. Three of those layers are the user interface layer, software logic layer and data access layer, which are inside the software. The fourth layer is a storage layer and the layer is outside the software.

The software has been programmed by using the C# programming language and Visual Studio 2008 has been used as a programming tool. The testing of the software has been done by using Visual Studio's test project.

As the result of design and implementation was achieved working software, even though some of the customer requirements were not implemented.

Keywords	configuration tool, fourth layered architecture, object oriented, NET framework

Sisällys

Tiivistelmä

Abstract

Lyhenteet

1 Johdanto 6

2 Graafinen konfigurointityökalu 7

2.1 Järjestelmän toiminta 7
2.1.1 Ohjelmistokehys 9
2.1.2 Ohjelmiston komponentit 14
2.2 Reitittimet kohdelaitteina 23
3 Ohjelmistovaatimukset 27

3.1 Ohjelmistotuotanto 27
3.2 Toiminnalliset vaatimukset 28
4 Järjestelmänsuunnittelu 31

4.1 Olioperustaisuus 31
4.2 Ohjelman suunnittelu 33
4.3 Käyttöliittymäkerros 34
4.4 Sovelluslogiikka-, tiedonsaanti- ja säilytyskerros 37
5 Toteutus ja testaus 39

5.1 Tekninen toteutus 39
5.2 Ohjelman testaus 41
6 Jatkokehitysmahdollisuudet 42

7 Yhteenveto 43

Lähteet

Liite 1: Vaatimustaulukko 46
Liite 2: Luokkakaavio 49

Lyhenteet

API	*Application Programming Interface*, sovellusohjelmointiliittymä on joukko tietorakenteita, luokkia ja metodeja.
BGP	*Border Gateway Protcol*, autonomisten alueiden reititysprotokolla. Toimii reitititysprotokollana isoissa verkoissa.
CLI	*Common Language Infrastructure*, yleiskielinen infrastruktuuri. Sisältää .NET-ohjelmistokehyksen määritykset.
CLR	*Common Language Runtime*, ohjelmointikielille yhteinen ajoympäristö.
DLL	*Dynamic-Linked Library*, dynaaminen linkki kirjasto ja ei-ajettava ohjelmatiedosto.
FCL	*Framework Class Library*, .NET-ohjelmistokehykselle määritetyt luokkakirjastot.
GCT	*Graphical Configuration Tool*, graafinen konfigurointityökalu. Käyttöliittymä reitittimien konfigurointiin.
GUI	*Graphical User Interface*, graafinen käyttöliittymä. Graafinen käyttöliittymä koostuu ikkunoista, painikkeista ja tekstikentistä.
IL	*Intermediate Language*, välikieli ja tavukoodillinen CLR-ajoympäristön kieli.
IOS	*Internetwork Operating System*, sisäverkkokäyttöjärjestelmä. Toimii käyttöjärjestelmänä Cisco-reitittimissä.
JIT	*Just-In-Time*, ajonaikainen kääntäjä ja CLR-ajoympäristön osa.
OSPF	*Open Shortest Path First*, avoimiin standardeihin perustuva reititysprotokolla. Reitinvalinta perustuu lyhyemmän polun valintaan.
PE	*Portable Executable*, siirrettävä ja suoritettava kokoonpanon osa.
RIP	*Routing Information Protocol*, yleinen reititystietoprotokolla. Käytetään reitinmäärittämiseen tietoverkoissa.
SSH	*Secure Shell*, suojattu Shell-yhteys.
SQL	*Structured Query Language*, standardoitu relaatiotietokantojen kyselykieli. Sisältää joukon tietokantakyselyihin käytettyjä komentoja.
UML	*Unified Modeling Language*, yhdistetty mallintamiskieli. Mallintamis-standardi, joka sisältää joukon sääntöjä mallinnuskaavioiden tekemiseen.

1 Johdanto

Tässä insinöörikirjassa kerron graafisesta konfigurointityökaluohjelmasta, jonka tein opinnäytetyönä. Työlle asetettu tavoite oli suunnitella ja toteuttaa graafinen konfigurointityökalu, joka on tarkoitettu asetusten tekemiseen reitittimelle eli reitittimen konfigurointiin.

Työn idea sai alkunsa vuodelta 2007, jolloin suoritin tietotekniikan koulutusohjelman tietoverkot-suuntautumisvaihtoehdon laboratoriotöitä. Laboratoriotöissä verkkolaitteiden asetuksia tehtiin pitkillä ja vaikeasti muistettavilla tekstipohjaisilla komennoilla. Näistä laboratoriotöitä vaikeuttavista asioista syntyi idea kehittää ohjelma, jolla asetusten tekeminen voidaan tehdä helpommin. Ideasta muodostui tarve kehittää graafinen konfigurointityökalu (engl. *Graphical Configuration Tool*), jossa reitittimien asetusten tekeminen suoritetaan graafisesti, kun taas komentosarjakielisissä käyttöliittymissä se suoritetaan tekstuaalisesti. Konfigurointityökalussa helppokäyttöisyys on saavutettu nappeja ja parametrikenttiä apuna käyttäen.

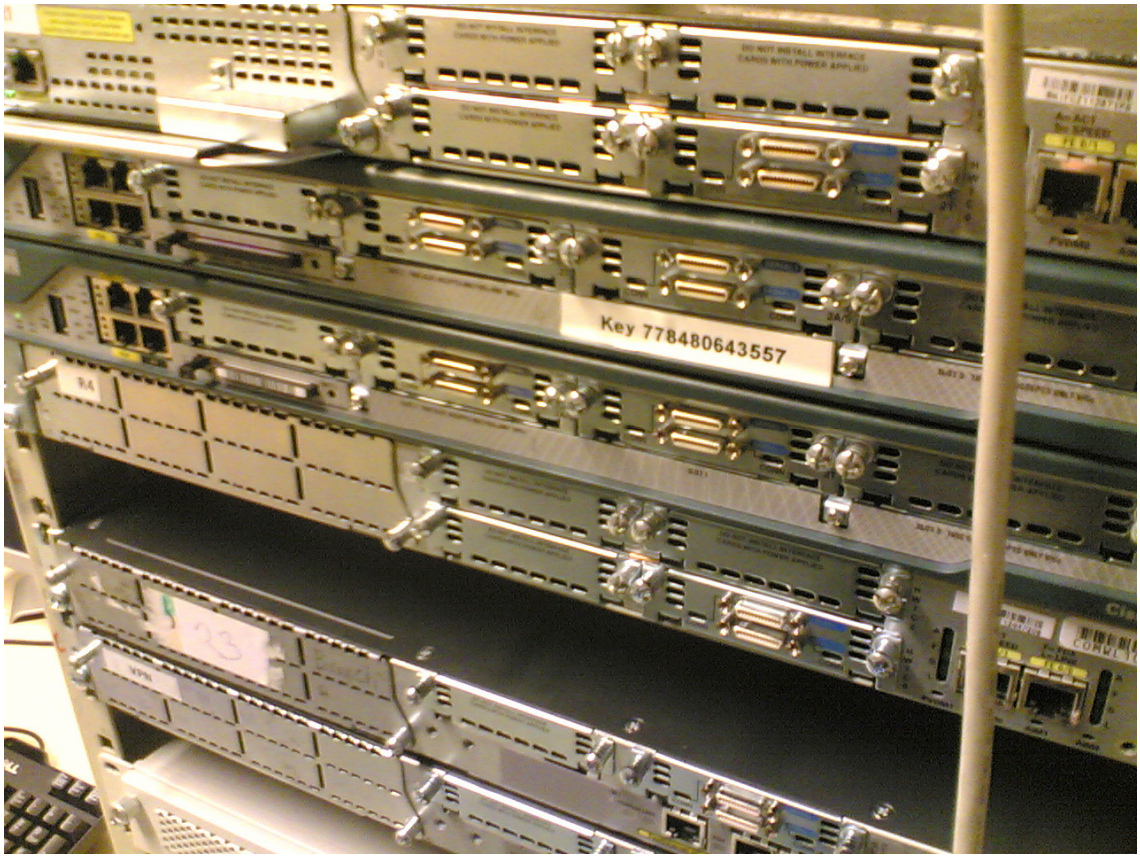
Työn tilaajana toimii Metropolia Ammattikorkeakoulu, ja ohjelmaa tullaan käyttämään apuna opetustyössä Metropolia Ammattikorkeakoulun Espoon toimipisteen tietoverkkolaboratoriossa. Tietoverkkolaboratorioissa opetetaan dataverkkojen suunnittelua, ohjelmointia ja ylläpitotehtäviä. Graafista konfigurointityökalua tullaan käyttämään erityisesti reitittimien ohjelmoinnin opetukseen.

2 Graafinen konfigurointityökalu

2.1 Järjestelmän toiminta

Järjestelmän pääkomponentit ovat graafinen konfigurointityökalu ja reitittimen käyttöjärjestelmä. Nämä kaksi pääkomponenttia kommunikoivat keskenään sarjaliikenneyhteyden kautta. Siten järjestelmän toiminta on yksiselitteisesti kahden erityyppisen käyttöliittymän välistä keskustelua. Järjestelmä koostuu käyttöympäristöstä, ajettavasta ohjelmasta, tietokannoista, ulkoisesta tiedostosta, kohdelaitteesta, kohdelaitteen käyttöjärjestelmästä ja kohdelaitteen käyttöjärjestelmän käyttöliittymästä sekä tietokoneen ja kohdelaitteen välisestä yhteydestä.

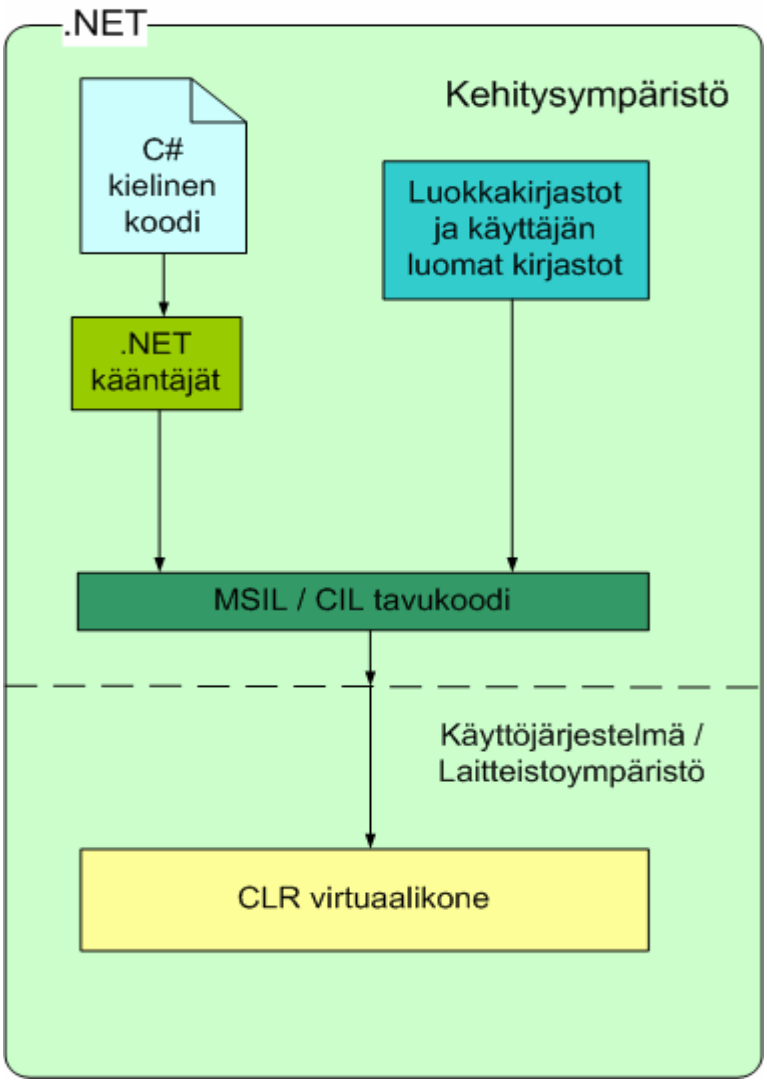
Ohjelmalla voidaan tehdä asetusten muutoksia suoraan reitittimelle tai reitittimen kautta siihen liitettyihin toisiin reitittimiin. Seuraavalla sivulla olevassa kuvassa 1 on esitetty työssä käytetty reititin, tuotenimeltään Cisco 2800 series. Kuvassa Cisco 2800 -sarjan reititin on kolmas alhaalta. Reitittimet on asetettu laitetelineeseen eli räkkiin. Reitittimet on sijoitettu reitittimen takaosa eteenpäin, koska reitittimien takaosassa on porttiliittymät, joihin johdot ovat liitettävissä. Seuraavissa alaluvuissa järjestelmän toimintaa kuvataan tarkemmin.



Kuva 1. Kohdelaiteena toimiva reititin, Cisco 2800 -sarja. Reititin näkyy kuvassa kolmantena alhaalta ylöspäin kulkiessa. Toiset kuvassa näkyvät laitteet ovat myös reitittimiä, Cisco 2600 -sarjasta. Kuva on otettu Metropolian tietoverkkolaboratoriossa.

2.1.1 Ohjelmistokehys

Graafisen konfigurointityökaluohjelman kehitys on suoritettu Microsoftin kehittämässä .NET framework -kehitysympäristössä. Kehitysympäristö on ohjelmistokehys, joka on saatavissa uudempiin Windows-käyttöjärjestelmiin. Richter (1, s. xxi) mainitsee ohjelmistokehyksen koostuvan kahdesta osasta, joita ovat ohjelmointikielille yhteinen CLR-ajoympäristö ja FCL-luokkakirjastot. CLR-ajoympräristöä kutsutaan myös virtuaalikoneeksi. Kuvassa 2 esitetään .NET framework -kehitysympäristön tärkeimmät osat.



Kuva 2. .NET framework -kehitysympäristö.

Ohjelman lähdekoodi voi olla mikä tahansa .NET framework 3.5 -kehitysympäristön nykyisin tukemista muutamasta kymmenestä ohjelmointikielestä. Käytetyimpiä ohjelmointikieliä ovat C# ja Visual Basic eli VB .NET. Lähdekoodi kirjoitetaan ihmiselle (ohjelmoijalle) ymmärrettävään muotoon ohjelman kehitysvaiheessa. Ohjelmointia on kuitenkin rajoitettu sillä tavalla, että ohjelmoijalla on käytettävissä ennalta määrättyjä sanoja, joita hän voi käyttää ohjelmakoodin rakentamiseen.

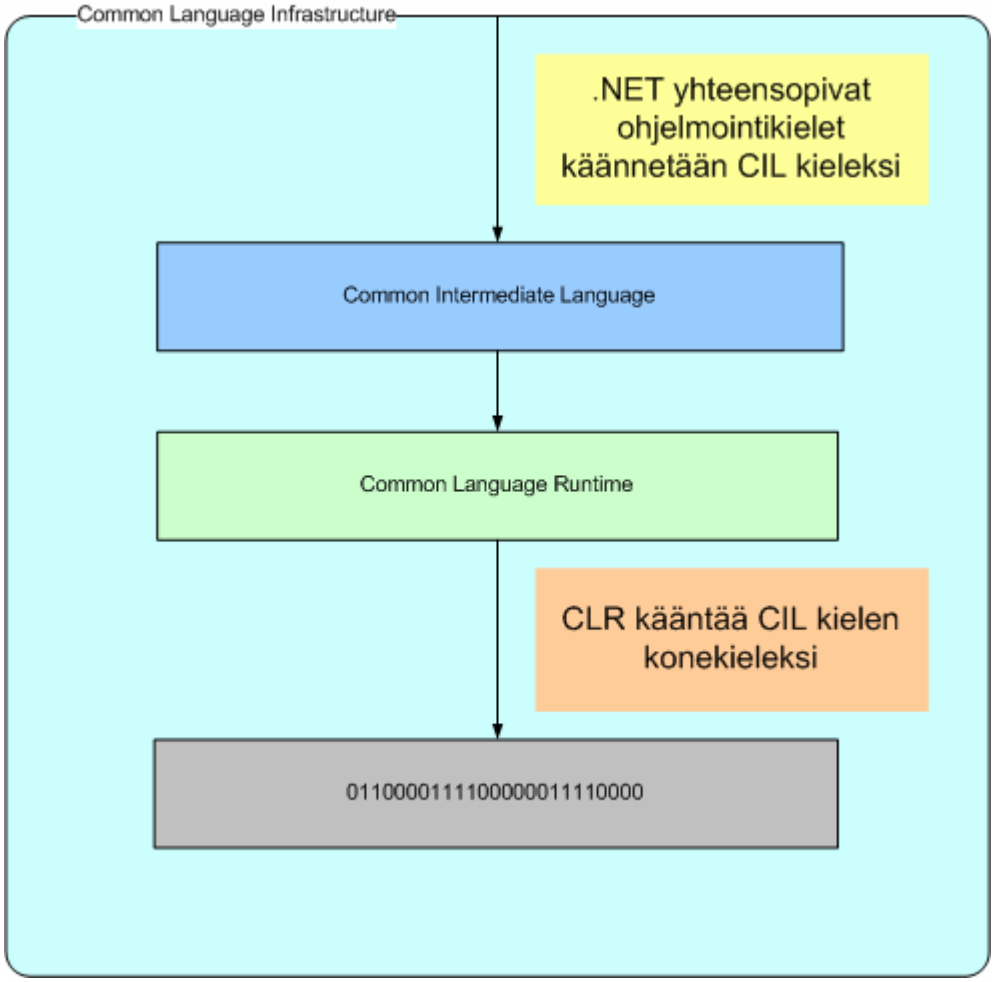
Ohjelman kehittäjä valitsee ohjelmassa käytettävät luokkakirjastot. Luokkakirjastot ovat lähdekoodiin liitettyjä nimiavaruuksia, joilla viitataan valmistettuihin luokkiin. Tavallisesti järjestelmän nimiavaruus viite on System-alkuinen ja se otetaan käyttöön using-komennolla. Luokkakirjastot ovat valmisluokkia, jotka tarjoavat tarvittavia luokkia ja metodeja, esimerkiksi tietokannan käsittelyyn ja yhteyden muodostamiseen. Troelsenin (2, s. 7) nimeää kehitysympäristön tarjoamiksi perusluokkakirjastoiksi tietokantayhteys-, työpöytä- GUI API-, turvallisuus-, etäyhteys-API-, säie-, tiedosto-I/O- ja Web API -luokkakirjastot. Ohjelmoijalla on mahdollisuus tehdä myös omia luokkakirjastoja.

.NET-kääntäjä, joka tämän ohjelman tapauksessa on C#-kääntäjä, kääntää lähdekoodin ja luokkakirjastot hallituksi moduuliksi tai toiselta nimeltään .NET-moduuliksi, jota tässä työssä käytän. Käännöksen aikana kehitysympäristö tarkistaa lähdekoodin oikeellisuuden ja ilmoittaa tarvittavista korjauksista ohjelmoijalle.

Hallittu moduuli on suoritusvalmiskokoonpano virtuaalikoneelle. Richter (1, s. 5) mainitsee hallitun moduulin osien olevan PE-otsikko, CLR-otsikko, metatiedot ja IL-välikoodi. Seuraavassa luvussa kerrotaan tarkemmin moduulin sisällöstä. Kokoonpano voi sisältää yhden tai useamman moduulin. Kokoonpano on ajettava sovellusohjelma (exe-tyyppinen tiedosto) tai DLL-tiedosto.

PE-otsikko on siirrettävä ja suoritettava Windows-ohjelmatiedosto. Otsikosta selviää tiedoston tyyppi, joka voi olla GUI, CUI tai DLL, sekä tiedoston luontiaika. CLR-otsikko sisältää tietoa moduulin hallittavuudesta sekä CLR-versiotietoa, jota tarvitaan tunnistamaan tuettu virtuaalikoneversio. CLR-otsikossa on myös tietoa main-metodin aloituskohdasta ja resurssiluettelosta sekä metatiedon sijainti- ja kokotietoja. Metatiedot ovat luokkatietoja tai toisin sanottuna taulukoita lähdekoodissa määritellyistä tyypeistä ja jäsenistä sekä taulukoista, joihin lähdekoodissa viitataan. Metatiedoilla poistetaan vanha tapa käyttää otsikko- ja kirjastotiedostoja. IL-välikieli on tavukoodi, jota nykyään kutsutaan yleiseksi välikieleksi (Common Intermediate Language, CIL) tai vanhalta nimeltään Microsoftin välikieleksi (Microsoft Intermediate Language, MSIL). Tämä välikieli syntyy lähdekoodin käännöksen tuloksena. Myöhemmin virtuaalikone kääntää tämän välikoodin prosessorin ymmärtämäksi konekieleksi. Konekielet ovat binäärimuotoisia tietoja. (1, s. 5–20; 3; 4.)

Yleiskielinen CLI-infrastruktuuri on avoin määritelmä, joka sisältää yleistyyppijärjestelmän, yleiskielimääritykset, välikielen ja virtuaalikoneen, joka toiselta nimeltään on virtuaalinen suoritusjärjestelmä (Virtual Execution System, VES). Yleiskieli infrastruktuuristandardissa ECMA-335 on infrastruktuurille mainittu seuraavia määrityksiä. Yleistyyppijärjestelmä tarjoaa järjestelmän, joka sisältää useita tyyppejä ja operaatioita, joita on löydettävissä monista ohjelmointikielistä. Välikieli kertoo ja viittaa tyyppeihin, jotka ovat määritetty yleistyyppijärjestelmässä. Yleiskielimääritykset sisältävät sääntöjä yleiskielisten määritysten osajoukoille ja käyttömuunnoksille. Virtuaalikone hyödyntää yleiskielijärjestelmää. Sen tehtävänä on ladata ja suorittaa yleiskieli-infrastruktuuriyhteensopivat ohjelmat. (5, s. 9; 6; 7; 8; 9.) Suoritettava ohjelma välitetään virtuaalikoneelle kokoonpanoina, jotka sisältävät yhden tai useamman moduulin. Virtuaalikoneen yhteydessä on ajonaikainen JIT-kääntäjä, joka suorittaa välikielen kääntämisen prosessorikohtaiseksi konekieleksi. Virtuaalikone on ohjelmallisesti toteutettu ”ohjelmamoottori”, joka jäljittelee todellisen tietokoneen tapaa suorittaa ohjelmia. Kuvassa 3 on esitetty infrastruktuurin toiminta.



Kuva 3. Yleiskieli-infrastruktuuri.

Nykyinen versio ohjelmistokehyksestä, .NET framework 3.5 -kehitysympäristö, mahdollistaa useantyyppisten ohjelmien, kuten Windows-ohjelmistojen, Web-ohjelmistojen, Web service -komponenttien, Windows CE -ohjelmistojen, konsoli-ohjelmistojen, Service-ohjelmistojen ja Microsoft Office -tuotteisiin liittyvien ohjelmistojen kehittämisen. [10.]

Ohjelman toiminta ohjelmistokehyksessä

Kehitysvaiheessa esikäännetyn ohjelman tiedostot ovat välikielimuodossa. Jokaisen arkkitehtuurikerroksen tiedostot ovat omissa kokoonpanoissaan ja kansioissaan. Käyttöliittymäkerroksen tiedostoista on luotu exe-tiedosto, koska se on ohjelman aloituskohta. Sovelluslogiikkakerroksen, tiedonsaantikerroksen, rajapinta- ja apuluokkatiedostoista on käännetty DLL-tiedostot.

Välikielitiedostoja ei ole vielä tässä vaiheessa tunnistettu tai kohdistettu käytettäväksi tietylle keskusyksikölle. Tämä ominaisuus mahdollistaa tavukoodillisen konekielen käytön useissa eri keskusyksiköissä. Välikieli on alimmalla tasolla, ihmisen luettavissa, oleva ohjelmointikieli. Se on myös oliosuuntautunut ohjelmointikieli, joka mahdollistaa sen käytön oliosuunniteltujen ohjelmien toteutuksessa. Välikieleksi käännetty ohjelma pystytään suorittamaan kaikissa ympäristöissä, joissa .NET framework -ohjelmakehys on tuettu.

Käyttäjän käynnistäessä ohjelman exe-tiedostosta välikieliset tiedostot tallennetaan keskusyksikön pinoon, josta välikieliset tiedostot ovat yleiskielisen infrastruktuurin käytettävissä. Virtuaalikone suorittaa muistinhallintaan, muistin varaamiseen ja vapauttamiseen tarvittavat toimenpiteet. Virtuaalikoneen yhteydessä oleva ajonaikainen JIT-kääntäjä suorittaa välikielisten tiedostojen kääntämisen natiiviksi konekieleksi, binäärikieleksi, sekä sijoittaa sen heap-muistiin eli dynaamisesti varattavaan muistialueeseen.

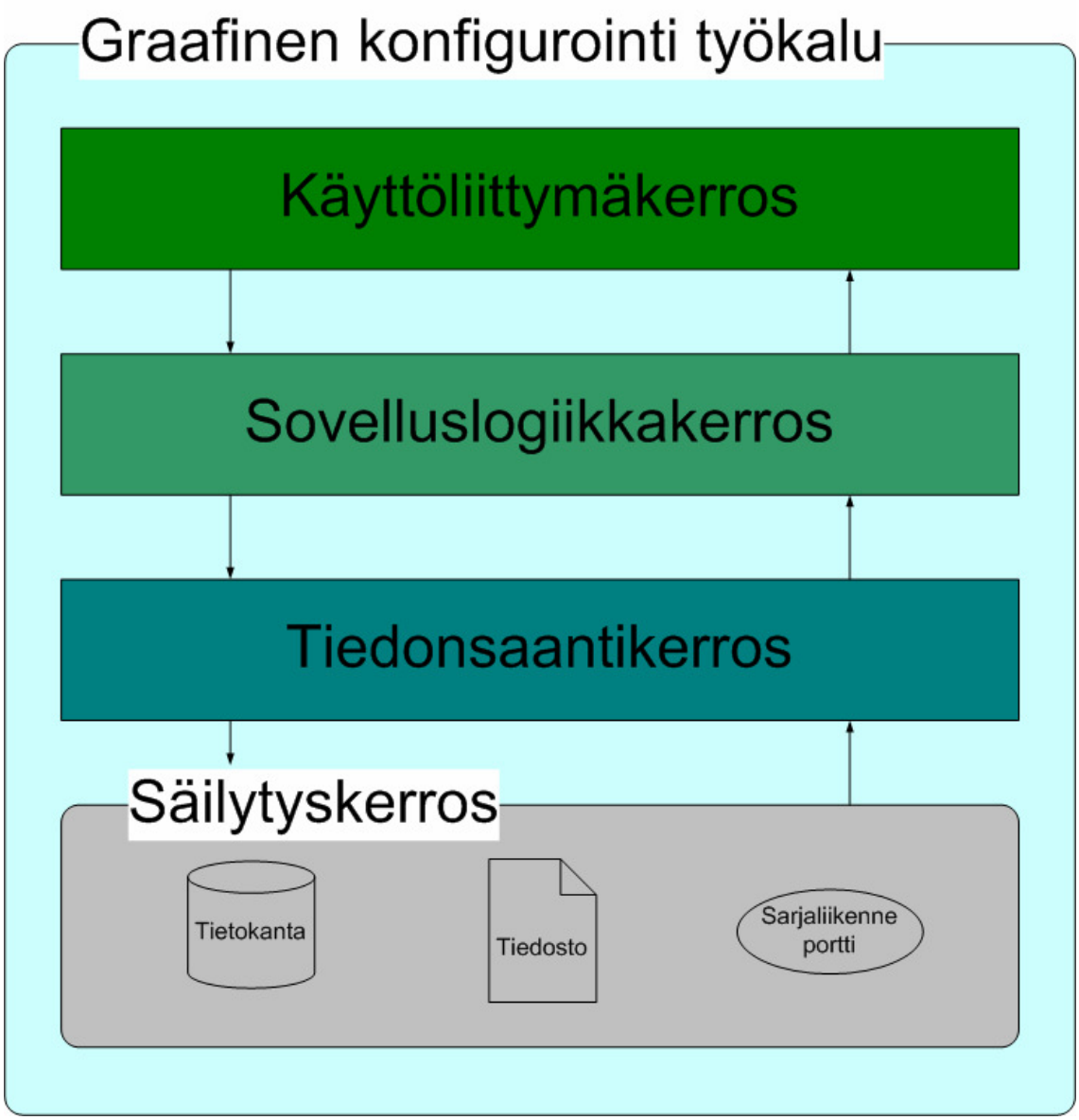
2.1.2 Ohjelmiston komponentit

Ajossa oleva ohjelma on koko järjestelmän tärkein osa. Se on alijärjestelmä, josta ohjelmoidaan järjestelmän kohdelaitetta, suoritetaan järjestelmän kommunikointi kohdelaitteeseen ja ohjataan kommunikointikanavaa. Ohjelman suunnittelussa on käytetty nelikerrosarkkitehtuuria. Ohjelman kolme ylintä kerrosta ovat itse ohjelma. Niissä samanlaisia tehtäviä suorittavat komponentit on sijoitettu omiin kerroksiin. Komponenttien tehtävät ovat toimia näkyminä, kontrollereina ja malleina. Neljäs kerros ei kuulu itse ohjelmaan, mutta on osa tätä järjestelmää. Neljännessä kerroksessa eli fyysisessä kerroksessa sijaitsevat kiintolevyllä oleva tiedosto ja tietokannat sekä tiedonsiirtokanava ja sarjaliikenneportti. Näkymien, kontrollerien ja mallien välisestä kommunikoinnista huolehtivat rajapinnat ja apuluokat, jotka ovat erilliskomponentteja eli ulkopuolisia komponentteja mutta ohjelma-alijärjestelmään kuuluvia osia.

Ohjelman arkkitehtuurityyli

Nelikerrosarkkitehtuuri on ohjelmistoarkkitehtuurityyli, jossa käyttöliittymät (näkymät) erotellaan muista sovellusaluetiedostoista, kuten kontrolleista ja malleista, sekä fyysisestä tasosta. Kerrosarkkitehtuuri koostuu tasoista, jotka on järjestetty jonkin abstrahointiperiaatteen mukaan nousevaan järjestykseen. Abstrahointiperiaatteena voi olla esimerkiksi laite-ihminen-skaalaus. Laitepäässä olevat tasot ovat matalammalla tasolla kuin ihmistä lähellä olevat tasot. Kerrosarkkitehtuurissa voi tapahtua ohituksia, eli ylin kerros voi kommunikoida suoraan alempien kerrosten kanssa hypäten välikerrosten yli. Arkkitehtuurityylejä käytetään apuna järjestelmän komponenttien ryhmittelyssä ja koko järjestelmän hahmottamisessa. Koskimies (11, s. 125) mainitsee tärkeimmiksi ryhmittelyyn käytetyiksi arkkitehtuurityyleiksi kerros- ja tietovuoarkkitehtuurit. Näistä erityisesti kerrosarkkitehtuuria voidaan käyttää lähes minkä tahansa järjestelmän kuvaamiseen.

Kuvassa 4 on esitetty järjestelmän ohitukseton nelikerrosarkkitehtuuri. Ylin kerros on käyttöliittymäkerros, joka on lähimpänä käyttäjää ja suoraan käyttäjän hallittavissa. Seuraavassa kerroksessa sijaitsee ohjelman logiikka. Kolmas ja alin ohjelman kerros on mallien kerros. Neljäs kerros ei ole itse ohjelmassa, mutta se on suoraan ohjelman hallittavissa. Neljännessä kerroksessa sijaitsevat järjestelmän fyysiset osuudet, kuten tietokannat ja tiedostot. (12, s. 152.)



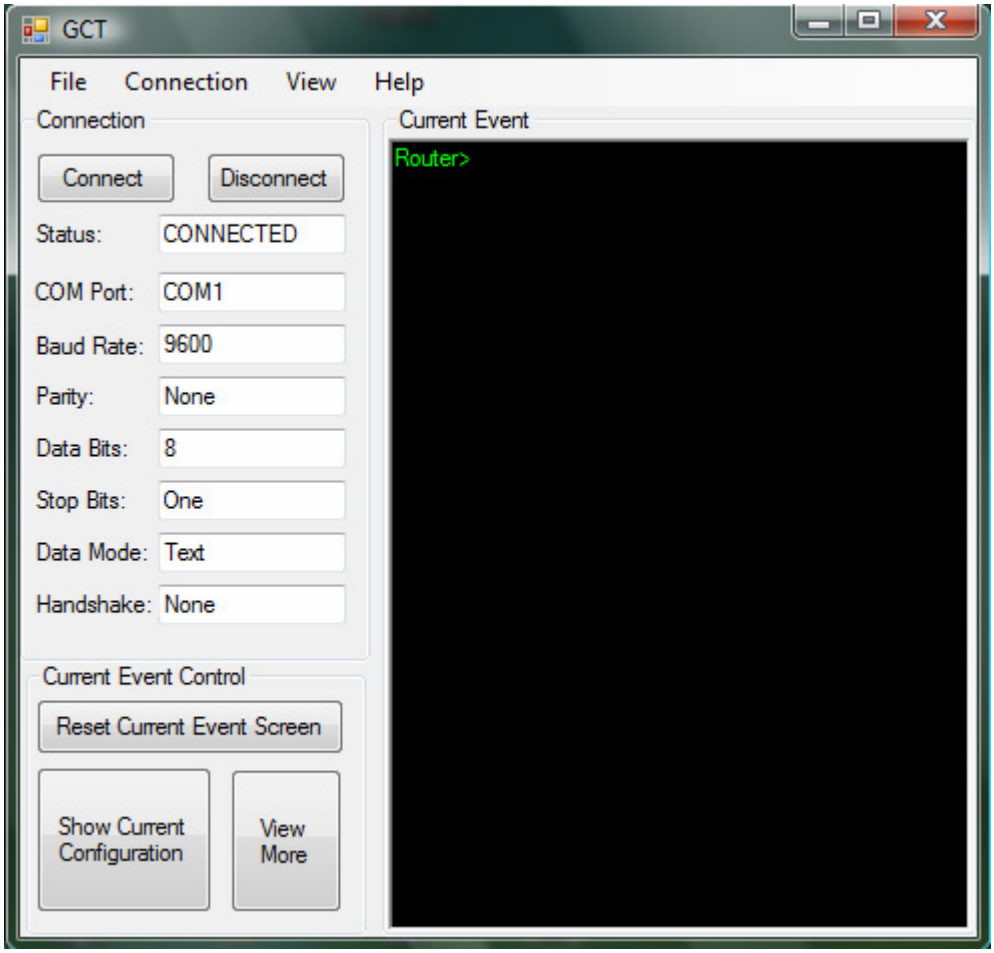
Kuva 4. Ohitukseton nelikerrosarkkitehtuuri.

Näkymät

Näkymät (View) ovat ikkunoita, jotka koostuvat valintanapeista ja valintalistoista sekä tekstinsyöttö- ja tekstinnäyttökentistä. Ne sijaitsevat kerrosarkkitehtuurissa ylimmässä eli käyttöliittymäkerroksessa tai toiselta nimeltään esityskerroksessa. Näkymien tehtävänä on ilmoittaa käyttäjälle ohjelman ja reitittimien asetuksien tila sekä mahdollistaa käyttäjälle kohdelaitteen hallinta. Näkymät toimivat niin, että jokaisella näkymällä on oma kontrolleri ja malli. Näkymässä suoritetut toiminnat ohjataan kontrollereille metodikutsuilla ja ominaisuuksien muutoksilla rajapintojen ja apuluokkien kautta. Näkymän päivittäminen tapahtuu kontrollerin käskystä. Kontrolleri syöttää uuden arvon ja näkymä esittää sen tekstikentässä. Seuraavana on kerrottu ohjelman sisältämien näkymien tarkoituksista. Ohjelmassa käytetyt näkymät ovat baudinopeus-, klassinen konfigurointi-, kohdelaitenimi-, komento-, MPLS-, porttiliittymä-, pääikkuna-, reititysasetus-, salasana-, sarjaliikenneportti-, telnet-yhteys-, tietokannan päivitys- ja VPN-näkymä.

Pääikkunanäkymä

Ohjelman pääikkunanäkymä (GCT) on näkymä, josta suoritetaan muiden ikkunoiden esilletuominen. Ikkuna tulee esiin ensimmäisenä ohjelmaa käynnistettäessä. Ikkunasta suoritetaan sarjaliikenneportin yhteyden avaaminen ja sulkeminen. Lisäksi ikkunasta voidaan muokata sarjaliikenneporttien asetuksia halutunlaiseksi. Jos asetuksia ei muuteta käyttäjän toimesta, oletusasetukset tulevat voimaan. Pääikkunassa olevan Current Event -näyttökentän tarkoitus on ilmoittaa käyttäjälle juuri suoritettu tapahtuma kohdelaitteen käyttöjärjestelmässä. Sarjaliikenneportille asetetut asetukset esitetään käyttäjälle tekstiruuduissa. Käyttäjä voi kutsua nykyisen konfiguraation kohdelaitteesta. Pääikkunan kautta käyttäjä pääsee hallitsemaan konfigurointitiedoston muistiin tallentamista ja muistista lukemista. Kuvassa 5 on esitetty ohjelman pääikkuna.



Kuva 5. Pääikkunanäkymä. Ikkunassa yhteys on avattu COM 1 -porttiin käyttäen oletusasetuksilla, ja reititin on ensimmäisellä komentotasolla.

Pääikkunasta suoraan avattavat näkymät

Suorittaessaan reitittimen asetusten tekemistä käyttäjä voi avata pääikkunasta seuraavia reitittimen asetusten tekemistä ja yhteyden muodostamista avustavia näkymiä, joista asetusten tekemistä avustavia näkymiä ovat klassinen konfigurointi-, komento-, tietokannan päivitys- ja telnet-yhteysnäkymät, sekä yhteyden muodostamista avustavia näkymiä ovat baudinopeus- ja sarjaliikenneporttinäkymät. Asetusten tekemistä avustavilla näkymillä suoritetaan itse reitittimen asetusten tekeminen ja yhteyden muodostamista avustavia näkymiä käytetään vain sarjaliikenneportin avaamisvaiheessa.

Baudinopeusnäkymä (Baud Rate View) on näkymä, josta käyttäjä syöttää baudinopeuden arvon, jota ei ole pääikkunan baudinopeuden valintalistassa. Baudinopeus tarkoittaa tiedonsiirtonopeutta bitti per sekunti.

Klassinen konfigurointi -näkymä (Classic Configuration View) on näkymä, jossa käyttäjä voi syöttää tekstityyppisiä komentoja. Näkymä on vaihtoehto graafisille asetusten tekemisen näkymille.

Komentonäkymä (Command View) on komentojenhallintanäkymä, josta käyttäjä valitsee ja muokkaa reitittimellä suoritettavan komennon käyttäen apunaan tekstikenttiä ja painallusnappeja. Käyttäjä voi valita reitittimen siirtymisen käyttäjätilaan, asettaa itsevalitseman nimen kohdelaitteelle, asettaa itsevalitsemat salasanat, tehdä kohdelaitteen porttien asetusten muokkaamisen, asettaa reititysprotokollat, asettaa virtuaalisen yksityisen verkon, asettaa moniprotokollatuen ja tallentaa konfiguraation kohdelaitteen muistiin. Komentonäkymä toimii näkymänä, josta avataan uusia reitittimen asetusten tekemiseen tarkoitettuja näkymiä, joita ovat kohdelaitenimi-, MPLS-, porttiliittymä-, reititysasetus-, salasana- ja VPN-näkymät.

Sarjaliikenneporttinäkymä (COMPort View) on näkymä, josta käyttäjä syöttää portin nimen, jota ei ole pääikkunan porttinimivalintalistassa.

Tietokannan päivitysnäkymä (Database Update View) on näkymä, josta suoritetaan tietokannan tietojen lisääminen ja poistaminen. Avattaessa tietokannan päivitysnäkymä tulee esiin sisäänkirjautumisdialogi. Sisäänkirjautumisdialogin kautta käyttäjä oikeutetaan muuttamaan tietokannan tietoja.

Telnet-yhteysnäkymän kautta käyttäjä suorittaa telnet-yhteyden avaamisen syöttämällä kohdelaitteen IP-osoitteen tai ihmisen luettavissa olevan selkokielisen osoitteen. Telnet-yhteyden kautta käyttäjä voi tehdä reitittimen asetusten tekemistä komento- tai klassista konfigurointinäkymää käyttäen.

Muut pääikkunan näkymät

Pääikkunan kautta käyttäjä voi kutsua tiedoston avaus- ja tallennus-dialogeja sekä avustus- ja ohjelman tietoikkunoita. Tietokannan päivitysnäkymän avaaminen kutsuu kirjautumisnäkymän esille. Kirjautumisnäkymän kautta käyttäjä saa oikeudet muokata tietokantoja.

Komentonäkymän kautta avattavat näkymät

Komentonäkymän kautta avattavat näkymät ovat näkymiä, joita käytetään itse reitittimen asetusten tekemiseen. Näkymien käyttö edellyttää yhteyden olemista avattuna. Seuraavana on kerrottu näkymien tarkoituksesta.

Kohdelaiteen nimi -näkymä (Hostname View) on näkymä, jolla käyttäjä voi muuttaa kohdelaitteen nimeä. Kohdelaitteessa oletusnimi on Router eli reititin.

Moniprotokollanimivaihdin (Multiprotocol Label Switching View) eli MPLS-näkymä on näkymä, josta käyttäjä voi muuttaa moniprotokollan asetuksia.

Porttiliittymänäkymä (Interface View) on näkymä, josta syötetään kohdelaitteen porttien asetuksia, esimerkiksi Ethernet-portin asetuksia. Käyttäjä voi muuttaa IP- eli Internet-protokollan osoitteen ja sen aliverkkomaskin sekä portin numeron ja porttipaikan arvoja. Käyttäjä voi asettaa portin toimintaan tai pois toiminnasta sekä asettaa sille kellonopeuden eli todellisen tiedonsiirtonopeuden. Kellonopeus voidaan asettaa muutettaessa sarjaportin asetuksia.

Reititysasetusnäkymä (Routing Configuration View) on näkymä, jossa asetetaan reititysprotokolla kohdelaitteen porttiin ja muutetaan reititysprotokollan asetuksia. Käyttäjä valitsee reititysprotokollan ja muuttaa sen parametrillisia asetuksia.

Salasananäkymä (Password Request View) on näkymä, josta käyttäjä syöttää kohdelaitteelle asetettavan salasanan. Näkymä tulee esiin, jos kohdelaite kysyy salasanaa käyttäjältä ja käyttäjän siirtyessä kohdelaitteen käyttöjärjestelmässä konfigurointitilaan. Salasana voidaan asettaa konsoli- ja virtuaaliterminaaliporteille.

Virtuaalisen yksityisen verkon näkymä (Virtual Private Network View) eli VPN-näkymä on näkymä, josta muutetaan virtuaalisen yksityisen verkon asetuksia.

Kontrollerit

Kontrollerit (Controller) eli ohjaimet toimivat siltana näkymien ja mallien välillä. Ne sijaitsevat nelikerrosarkkitehtuurissa toiseksi ylimmäisessä kerroksessa, eli sovelluslogiikkakerroksessa tai liiketoimintakerroksessa. Kontrollereiden tehtävä on reagoida järjestelmässä tapahtuneisiin muutoksiin, minkä ne voivat tehdä kahdella tavalla. Tavassa 1 kontrolleri saa herätteen mallissa tapahtuneesta muutoksesta ja päivittää näkymän vastaamaan uutta tilaa tai johtaa viestin seuraavalle kontrollerille. Toisen tavan mukaan kontrolleri reagoi näkymältä tulleeseen tapahtumaan ja muokkaa näkymästä tulevan tiedon mallille sopivaan muotoon sekä päivittää mallin. Kontrollereiden toiminta perustuu ominaisuuksien muutoksiin, tapahtumiin ja metodikutsuihin. Kommunikointi kontrollerista näkymiin ja malleihin suoritetaan rajapintojen ja apuluokkien avulla. Kontrollerit ovat järjestelmän komponenteista ainoita, jotka voivat kommunikoida keskenään toisten kontrollereiden kanssa. Mallit eivät voi kommunikoida keskenään, eivätkä näkymät keskenään.

Mallit

Mallit (Model) toimivat ajonaikaisina tiedonsäilyttäjinä. Ajon alkaessa ne sisältävät oletusasetukset, joita ohjelma tarvitsee toimiakseen. Ne sijaitsevat nelikerrosarkkitehtuurin kolmannessa kerroksessa eli tiedonsaantikerroksessa. Kontrollerit päivittävät mallien tietoja ja vastaavat mallien muutoksien välittämisestä näkymille. Ajonaikaiset tiedot on tallennettu tiedonsaantikerroksen luokkien ominaisuustietoihin. Mallit ovat suorassa yhteydessä fyysiseen kerrokseen eli säilytyskerrokseen, ja ne käsittelevät sieltä tulevaa tietoa kontrollereille. Erikoistehtäviä suorittavat mallit ovat porttimalli, tietokantamalli ja tiedostomalli. Seuraavaksi kerrotaan erikoistehtäviä suorittavista malleista.

Erikoistehtäviä suorittavista malleista porttimalli (Port Model) suorittaa yhteyden luonnin sarjaliikenneporttiin, portin yhteyden katkaisemisen, tiedon kirjoittamisen porttiin ja tiedon lukemisen portista. Tietokantamalli (Database Model) tarjoaa pääsyn tietokantoihin. Tietokannan käsittelyyn tarvittavat SQL-komennot suoritetaan tietokantamallissa. Mallista kirjoitetaan kysely tietokantaan ja tietokannasta luetaan arvoja malliin. Tiedostomallin (File Model) kautta suoritetaan tiedostoon tallentaminen ja tiedostosta lukeminen. Tiedostoa käytetään käytössä olevan konfiguraation tallentamiseen ja vanhan konfiguraation ajamiseen kohdelaitteelle.

Rajapintaluokat

Rajapintaluokat (Interface) toimivat kerrosten välillä tarjoten ominaisuuksia ja virtuaalisia metodeja kahden eri kerroksen luokkien perittäviksi. Niissä on määritetty ainoastaan metodien ja ominaisuuksien kutsutapa. Itse ominaisuudet ja virtuaaliset metodit toteutetaan rajapintoja käyttävissä luokissa. Ominaisuudet ovat alustamattomia jäseniä eli muuttujia, joille on toteutettu välikielen vakioiksi määrittämät get- ja set-metodit. Virtuaaliset metodit ovat operaattoreita eli funktioita, joiden toteutus voi olla erilainen eri luokissa. (1, s. 325.) Ohjelmassa rajapinnat on jaettu kahteen ominaisuuksia ja metodeja sisältävään luokkaan, jotka tekevät koodista selvemmin luettavan. Rajapintojen vaikutuksesta kerrokset eivät ole suoraan yhdistettyinä toisiinsa, mikä tekee niistä helpommin vaihdettavat. Esimerkiksi käyttöliittymätasolla tehty ohjelmakoodin muutos ei vaikuta koko ohjelmaan, vaan tasot ovat päivitettävissä kerroksittain, eikä koko ohjelmaa tarvitse kääntää uudelleen. Rajapinnat on sijoitettu käyttöliittymäkerroksen ja sovelluslogiikkakerroksen väliin sekä sovelluslogiikka ja tiedonsaantikerroksen väliin.

Apuluokat

Apuluokat (Assistance Class) välittävät rajapintatyyppiset oliot kerroskomponenteille. Apuluokalla estetään uusien olioiden luominen new-metodilla käyttöliittymätasolla. Välittämällä viitteen seuraavan kerroksen oliosta käyttöliittymätasolle estetään kerroksien kiinteä liittäminen toisiinsa. Apuluokat toimivat rajapintojen yhteydessä käyttöliittymäkerroksen ja sovelluslogiikkakerroksen sekä sovelluslogiikkakerroksen ja tiedonsaantikerroksen välillä.

2.2 Reitittimet kohdelaitteina

Reitittimen käyttöjärjestelmän arkkitehtuuri on yksirakenteinen (monoliittinen), eli koko käyttöjärjestelmä on yhtenä tiedostona ja kaikki prosessit jakavat yhtä muistialuetta. Sisältä reititin on tavallisen tietokoneen tapainen. Reititin sisältää tiedonsiirtoväyliä tiedonsiirtoon, prosessori käskykannan suorittamiseen ja muistilohkoja tiedon tallentamiseen sekä Cisco-yhtiön kehittämä käyttöjärjestelmä (Cisco IOS). (13, s. 10.)

Reititin on tietoverkkojen yhteydessä käytetty verkkolaite. Sillä voidaan yhdistää tietoverkkoja toisiinsa, ja sen tehtävä on reitittää sille tulevat paketit toisille reitittimille eli välittää tietoa tietoverkkojen eri osien välillä tai välittää tietoa muuntyyppisille tietoverkkolaitteille. Poikkeavissa olosuhteissa reititin voi myös estää tiedonvälittämisen muille verkkolaitteille. Tästä suojausominaisuudesta käytetään nimitystä pääsylista (Access list).

Reititin välittää sille tulevan tiedon seuraaville siihen kytketyille laitteille. Suunnanmäärittämiseen se tarvitsee kuitenkin kohdelaitteen osoitteen. Osoitteina tietoverkoissa käytetään IP-osoitteita eli Internet-protokollaosoitteita. IP-osoitteista nykyisin käytössä oleva IPv4 (Internet-protokolla versio 4) on 32-bittinen binääriluku, joka kertoo laitteen osoitteen verkossa. Jokaisella verkossa olevalla laitteella tulee olla yksilöllinen osoite, jotta laite voidaan tunnistaa ja sille voidaan välittää tietoa ilman tiedon välittymistä väärään paikkaan. Uudempi Internet-protokolla on versiota 6 (IPv6), joka mahdollistaa suuremman määrän osoitteita ja verkkolaitteita tietoverkossa. Internet-protokollan päätehtävä on välittää paketteja paikasta toiseen. (14, s. 572–594.)

Reitittimet vaihtavat tietoa olemassaolostaan ja osoitteistaan reititysprotokollien avulla. Nykyreitittimille on tarjolla useita reititysprotokollia, kuten RIP ja OSPF. RIP-reititystietoprotokolla on yleinen reititysprotokolla, jota käytetään reitin laskemiseen tietoverkossa. Reitti määritetään laskemalla lähdelaitteen ja kohdelaitteen välissä olevien reitittimien määrällä. Toinen mainittava reititysprotokolla on OSPF (Open Shortest Path First) eli lyhimmän reitin valinta -protokolla. Tämän reititinprotokollan ominaisuus on valita lyhyempi reitti. (14, 731–736, 757–761.)

Reititin joutuu toimimaan erilaisten asetuksien alaisuudessa. Siten sen on oltava uudelleen ohjelmoitavissa suorittamaan haluttuja tehtäviä. Tämä on mahdollistettu tekemällä reitittimelle oma käyttöjärjestelmä, jonka kautta reitittimen ominaisuuksia voidaan muuttaa halutunlaisiksi. Ciscon reitittimissä on Cisco-yhtiön kehittämä IOS-käyttöjärjestelmä. Käyttöjärjestelmässä on yksinkertainen tekstipohjainen komentoliittymä Command Line Interface (CLI), jolla asetuksia voidaan muuttaa. Käyttäjä syöttää tietyn komentosanoista koostuvan lauseen, jossa voi olla myös numeroarvoja. Komentosanat ovat ennalta määrättyjä sanoja, jotka käyttöjärjestelmä tunnistaa ja pystyy suorittamaan reitittimen prosessorissa. Tietty sana vastaa tiettyä asetusta, jolla reitittimen ominaisuus muutetaan. Seuraavana esitetään esimerkki reitittimen nimen muuttamisesta. (14, s. 55–59; 15, s. 3.)

Käyttäjä avaa yhteyden reitittimeen sarjaliikenneportin kautta ja painaa enter- tai return-painiketta. Hyväksynnän seurauksena reititin tulostaa näkyviin komentokehotteen, joka on Cisco IOS -käyttöjärjestelmän ensimmäinen komentotaso. Ensimmäinen komentotaso on tila, joka tulee ensimmäisenä esille reitittimelle kirjautumisen jälkeen. Tässä tilassa käyttäjä pystyy tekemään peruskomentoja. Käyttäjä ei pysty tekemään reitittimen asetuksien tekemistä tässä tilassa. (14, s 55.)

Router>

Käyttäjä kirjoittaa enable-komennon ja suorittaa hyväksynnän siirtyäkseen oikeutettuun tilaan, jolloin reititin kysyy käyttäjältä salasanan, jos tällainen on asetettu tilaan pääsemisen ehdoksi. Oikeutetussa tilassa käyttäjä voi siirtyä reitittimen asetusten tekemiseen ja käynnistää järjestelmän [14, s. 55].

Router>enable
Password:
Router#

Oikeutettuun tilaan päästyään käyttäjä siirtyy konfigurointitilaan kirjoittamalla komennon configurate terminal.

Router#configurate terminal
Router(config)#

Konfigurointitilassa käyttäjä pystyy tekemään reitittimen asetuksia, kuten asettaa reitittimelle uuden oletusnimestä eroavan nimen. Uusi nimi annetaan kirjoittamalla hostname-komento ja sen perään haluttu reitittimen nimi.

Router(config)#hostname Oma
Oma(config)#

Konfigurointitilasta palataan oikeutettuun tilaan kirjoittamalla exit- tai end-komento.

Oma(config)#exit
Oma#

Tehdyt muutokset ovat voimassa vain ajonaikaisessa muistissa, Flash-muistissa. Muutettu asetus tulee myös tallentaa reitittimen pysyväismuistiin eli käynnistysmuistiin, muussa tapauksessa seuraavan käynnistyksen yhteydessä tehdasasetukset palautetaan eivätkä tehdyt muutokset enää ole muistissa. Tallentaminen suoritetaan kirjoittamalla copy-komento, jonka jälkeen annetaan lähde- ja kohdeosoite.

Oma#copy running-config startup-config

Nykyreitittimille on myös käytössä Web-pohjainen konfigurointimenetelmä, jonka kautta voidaan tehdä reitittimen asetuksia verkon kautta.

3 Ohjelmistovaatimukset

3.1 Ohjelmistotuotanto

Ohjelmistotuotanto jaetaan osa-alueisiin, joita ovat liiketoiminta tai johtaminen, laatujärjestelmä, hankkeiden hallinta tuotetasolla ja projektinhallinta. Projektinhallinta sisältää kehitysprosessin vaiheet, kuten määrittelyn, suunnittelun, ohjelmoinnin ja testauksen sekä tuotteen valmistumisen jälkeen käyttöönoton ja ylläpidon. Projektinhallinta sisältää myös tukitoimintoja, kuten tuotteenhallinnan, laadunvarmistuksen, dokumentoinnin ja vaatimustenhallinnan, jotka ovat voimassa koko ohjelman elinkaaren ajan eli ohjelmiston kehittämisestä sen poistamiseen käytöstä. (16, s. 35–36.) Tässä insinöörityössä käsitellään projektinhallinnan vaiheita ja niiden tukitoimintoja, jättäen muut ohjelmistotuotannon osa-alueet pois.

Haikala ja Märijärvi (16, s. 225) mainitsevat projektin olevan kertaluoteinen tehtävä, joka toteutetaan suunnitelmallisesti ennalta laaditun aikataulun mukaisesti. Lisäksi he mainitsevat projektilla olevan määritetyt resurssit ja organisaatio. Projektinhallinta on projektin osittamista, suunnittelemista ja seurantaa, riskien hallintaa ja työmäärien arviointia. Siihen liittyviä tehtäviä ovat projektin suunnittelu, käynnistäminen, toteutumisen seuranta ja ohjaus sekä projektin päättäminen.

Graafisen konfigurointityökalun kehityksessä on käytetty vaihejakomallityyppistä protoilumallia, jossa kehitys alkaa protoiluprojektista, joka sisältää vaiheet määrittely, suunnittelu, toteutus ja testaus. Tuloksena on saatu ohjelman prototyyppi, joka on pohjana seuraavalle projektille, tuoteprojektille, jossa itse ohjelma toteutetaan. Myös tuoteprojekti sisältää määrittely-, suunnittelu-, toteutus- ja testausvaiheet.

Kehitysprosessin määrittelyvaiheessa asiakkaan tekemät vaatimukset on analysoitu ja niistä on johdettu ohjelmistovaatimukset. Ohjelmistovaatimukset ovat toiminnallisia vaatimuksia, joista tehtyä dokumenttia kutsutaan toiminnalliseksi määrittelyksi. Tässä dokumentissa kuvataan ohjelmiston toiminnot, ei-toiminnalliset vaatimukset ja rajoitteet. Seuraava alaluku 3.2 käsittelee ohjelmiston toiminnalliset vaatimukset.

3.2 Toiminnalliset vaatimukset

Protoilumallin yhteydessä määritin ohjelmalle perusvaatimuksia, jotka ohjelman prototyypin tuli täyttää. Määritin ohjelman perusvaatimukset itse tietämieni tietoverkkolaitteiden asetusten tekemiseen liittyvien tarpeiden perusteella. Lisäksi sähköpostikeskustelun ja henkilökohtaisen keskustelun aikana Puskan kanssa (17; 18) lisävaatimuksiksi asetettiin kiinteä reititys, liitäntäparametrien lisäys, BGP-reititysprotokollan pois jättäminen ohjelmasta, ja Puskan ehdotuksesta ohjelma päätettiin kohdistuvan ainoastaan reitittimiin, eli kytkimien asetusten tekeminen päätettiin jättää pois. Sähköpostikeskustelussa Rahkosen kanssa (19) ohjelmaan päätettiin lisätä turvallinen Shell-konfigurointi eli SSH-konfigurointi ja moniprotokollanimenvaihdin (MPLS) sekä jättää pois kehysvälitin asetusten eli Frame Relay -asetusten tekeminen reitittimelle. Myöhemmässä henkilökohtaisessa keskustelussa Puskan kanssa (18) SSH-konfigurointi päätettiin jättää pois ohjelmasta.

Prototyyppi sisälsi käyttöliittymän esimuodon, kohdelaitteen ohjelmaan yhdistämiseen tarvittavat toiminnot ja muutamia toimintoja, kuten reitittimen konfiguraation tulostamisen käyttäjälle sekä klassisen tekstimuotoisen konfigurointimahdollisuuden. Seuraava versio oli lopullinen versio, ja se toteutettiin täyttämään kaikki jäljelle jääneet vaatimukset. Ohjelman protoiluvaiheen jälkeen suunnitelmasta poistettiin työn tilaajan ehdotuksesta muutamia vaatimuksia. Ohjelma suunniteltiin muuttamaan vain reitittimen asetuksia, ja alun perin suunniteltu verkkopiirtoikkuna poistettiin ohjelmasta tarpeettomana ominaisuutena. Verkkopiirtoikkuna oli suunniteltu alun perin kohdelaitteiden asetusten tekemisen avuksi suoritettaessa useamman laitteen asetusten muuttamista. Seuraavana kerrotaan ohjelmassa täytetyt vaatimukset.

Ohjelman vaatimukset

Perusvaatimuksina ohjelman on sallittava käyttäjän tehdä reitittimen asetuksia, kuten porttiliittymien asetuksia, kohdelaitteen nimen muuttaminen, IP-osoitteiden asettaminen kohdelaitteiden portteihin, jotta portteihin voi kohdistaa liikennettä ja jotta ne ovat verkon tunnistettavissa. Verkossa toimimisen edellytyksenä reitittimellä tulee olla reititysprotokolla. Käyttäjän tulee voida muuttaa reititysprotokolla halutunlaiseksi, lisäksi MPLS-asetus tulee olla tehtävissä reitittimelle.

Reitittimen asetusten suojaamiseksi reitittimen virtuaaliterminaali- ja konsoliyhteyksiä varten tulee reitittimelle voida asettaa selvätekstinen salasana tai suojattusalasana. Ohjelman tulee myös reagoida, jos reititin kysyy salasanaa käyttäjältä.

Näkymässä valitun komennon tulee siirtyä kohdelaitteelle reaaliajassa yhteyden niin salliessa. Ohjelman tulee tulostaa kohdelaitteen käyttöjärjestelmän tapahtumat käyttäjän nähtäville, ja käyttäjän suorittamia komentoja tulee näkyä samassa tapahtumaruudussa päänäkymässä. Tunneloidun yhteyden avaamiseksi reitittimelle pitää pystyä tekemään VPN-asetukset.

Kohdelaitteen kautta tulee voida tehdä muiden reitittimien asetuksia. Tämä ominaisuus on mahdollistettu lisäämällä reitittimelle telnet-yhteyden avaaminen muihin reitittimiin.

Liitteessä 1 olevassa taulukossa 1 on lueteltu kaikki työlle asetetut perusvaatimukset, selitykset ja niiden toteuttaminen.

Taulukossa 1 on esitetty asiakkaan esittämät lisävaatimukset, jotka päätettiin toteuttaa määrittelyvaiheessa.

Taulukko 1. Lisävaatimukset ohjelmalle.

Lisävaatimus	Selitys	Toteutettu / Ei toteutettu
LV1	SSH-konfigurointi	Ei toteutettu.
LV2	MPLS-konfigurointi	Toteutettu.
LV3	Kiinteä reititys	Toteutettu.

4 Järjestelmänsuunnittelu

4.1 Olioperustaisuus

Graafisen konfigurointityökalun suunnittelussa on käytetty olioperustaisia menetelmiä, kuten oliomäärittelyä ja oliosuunnittelua. Olioperustaisuuden mukaan reaalimaailman asioiden katsotaan koostuvan olioista. Oliosuuntautuneiden eli olioperustaisten menetelmien peruselementtinä on olio, jossa yhdistetään tietorakenne (jäsenet) ja käyttäytyminen (operaatiot). Koskimies (20, s. 9) mainitsee, että olio on määriteltävissä ympäristöstään erottuvana kokonaisuutena, jolla on oma identiteetti, sisäinen rakenne ja suhteet ympäristöön. Teknisesti ajatellen olio on ohjelman rakenteen perusyksikkö. Tiedon ja toimintojen kokoamista samaan rakenteeseen kutsutaan kapseloinniksi. Olio-ohjelmoinnin periaatteiden mukaan samanlaiset oliot ryhmitellään luokiksi.

Olioperustaisuus antaa mahdollisuuden systemaattiseen ohjelmistokehitykseen alkaen vaatimusmäärittelystä ja päättyen suoritettavaan koodiin. Olioperustaiseen suunnitteluun kuuluvia menetelmiä ovat oliomäärittely (Object Oriented Analysis, OOA) ja oliosuunnittelu (Object Oriented Design, OOD) [16, s. 349].

Oliomäärittelymenetelmä sisältää vaiheet asiakastarpeiden määrittelemisestä arkkitehtuurisuunnitteluun. Menetelmässä oliot tunnistetaan käyttötapauksista keräämällä substantiiveja suoraan tekstistä. Olioiden attribuutit ja metodit eli ominaisuudet ja käyttäytyminen sekä olioiden väliset yhteydet tunnistetaan myös käyttötapauksista. Määrittelyvaiheen mallintamisen suoritin UML-standardin mukaisten käyttötapauksien ja käyttötapauskaavion avulla.

Oliosuunnittelumenetelmä sisältää osaksi arkkitehtuurisuunnittelu- ja moduulisuunnitteluvaiheet. Menetelmässä oliomäärittelyvaiheen oliot, ominaisuudet ja käyttäytyminen sekä olioiden väliset suhteet tarkennetaan käyttäen mallinnuksessa apuna staattista ja dynaamista mallinnusta. Staattisessa mallinnuksessa käytin UML-standardin mukaisia luokkakaavioita eli tarkensin ohjelman luokkarakennetta. Staattisessa vaiheessa lisäsin myös käyttöliittymä- ja ohjausluokat eli näkymät ja kontrollerit. Dynaamisessa mallinnuksessa hyödynsin UML-standardin sekvenssikaavioita tarkentamaan luokkien toimintoja eli ryhmiteltyjen olioiden käyttäytymistä. Seuraavissa alaluvuissa 4.2–4.4 kerrotaan ohjelman suunnittelusta ja kerrosrakenteista tarkemmin.

4.2 Ohjelman suunnittelu

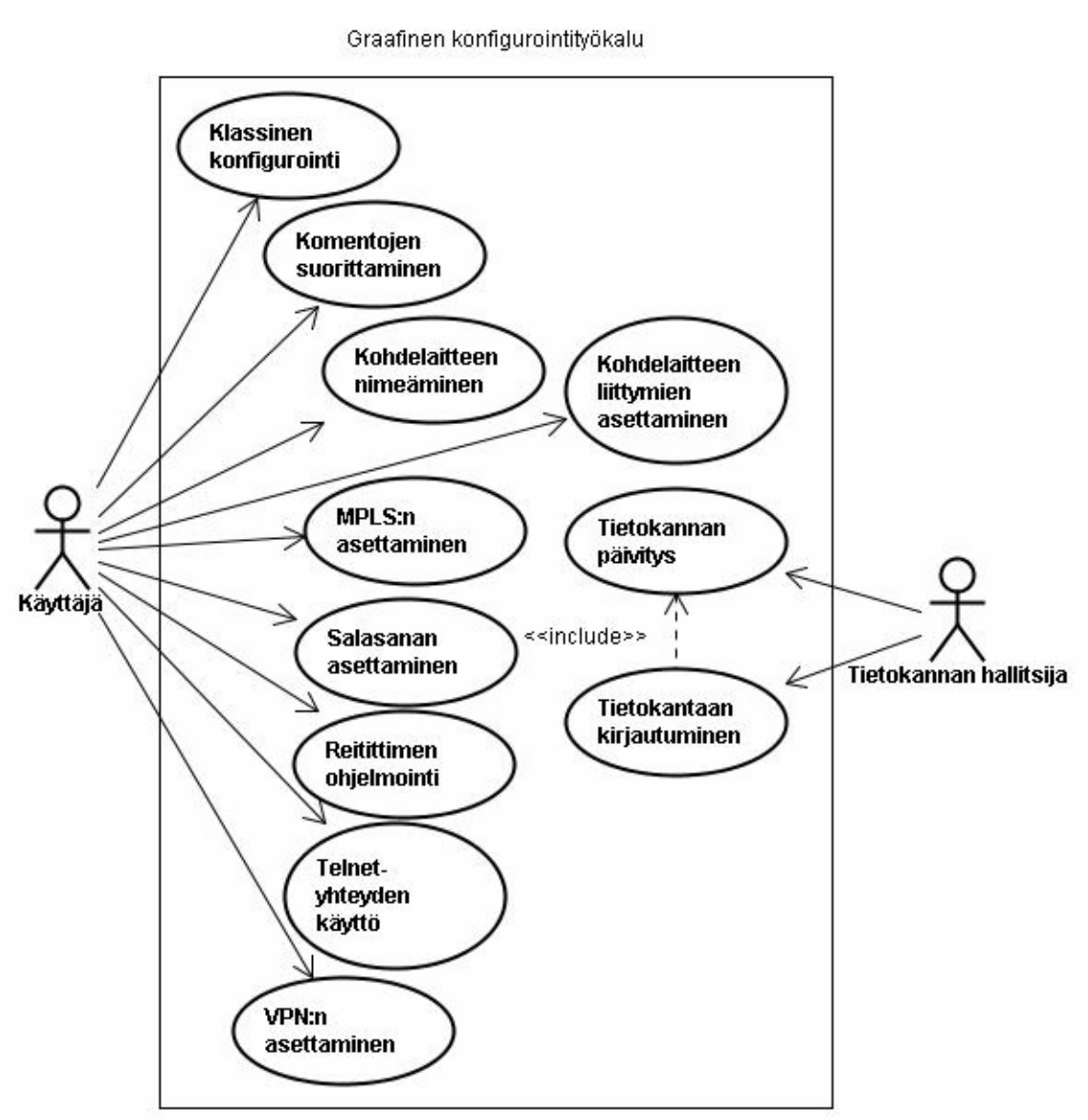
Graafisen konfigurointityökalun kehitys on suoritettu kahdessa vaiheessa. Suunnittelussa on käytetty ohjelmistokehitystyyppinä protoilumallia. Protoilumalli sisältää vaiheita, joissa ohjelma kehitetään. Ensimmäinen vaihe on prototyyppivaihe, jossa kehitin ohjelmasta prototyypin. Toisessa vaiheessa eli tuotevaiheessa kehitin itse ohjelman. Protoilumalleja käytetään yleensä vaatimusten tarkentamiseen asiakkaalle tai itse kehittäjälle. Siinä kehitetty ohjelma on yleensä puutteellinen, ja ainoastaan ohjelman päätoiminnot on toteutettu. (16, s. 42–44.)

Prototyyppivaiheessa suunnittelin ohjelmasta yksinkerroksisen, eli näkymät, kontrollerit ja mallit oli sijoitettu samaan kerrokseen. Prototyypissä kaikki reitittimen käyttöjärjestelmässä ajettavat komennot olivat tietokannassa, josta ohjelma haki ne suoritettavaksi.

Tuotevaiheessa näkymät, kontrollerit ja mallit oli jaettu omiin kerroksiin sekä kerrosten väliin oli asetettu rajapinnat ja apuluokat kerrosten välisen kommunikoinnin mahdollistamiseksi. Lisäksi ajattelin tiedoston, tietokannat ja sarjaliikenneportin kuuluvan omaan neljänteen eli fyysiseen kerrokseen. Prototyyppivaiheessa kaikki reitittimen käyttöjärjestelmän komennot oli sijoitettu yhteen tietokantaan, mikä aiheuttaa ongelmia ohjelman tietokannan päivitysvaiheessa. Ongelmana oli se, että lisättäessä uusia osia ohjelmaan ja päivitettäessä reitittimen komentoja sisältävää tietokantaa vastaamaan uusia osia joutuu tietokannan päivittäjä lisäämään useita komentorivejä tietokantaan, koska yksittäinen komentolause voi sisältää useita parametreja. Lisäksi parametreilla voi olla useita arvoja eli tietokannasta tulisi moniulotteinen tai useita rivejä sisältävä. Korjasin tämän päivitystä vaikeuttavan asian suunnittelemalla ohjelmalle useita komentotietokantoja parametrien arvoille ja sijoittamalla ohjelman osien yleensä käyttämät reitittimen käyttöjärjestelmän komennot ohjelman malleihin. Näin ohjelman päivittäminen tuli helpommaksi käyttäjälle ja sen kehittäjälle.

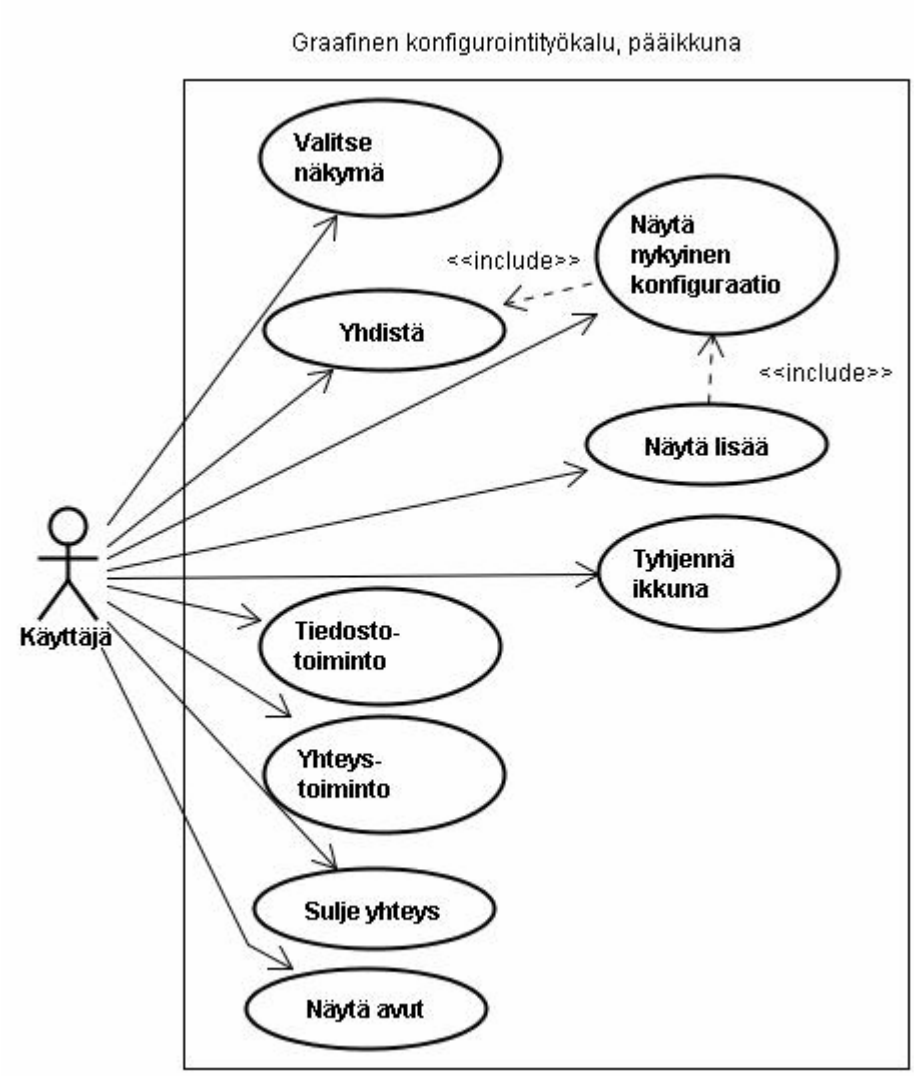
4.3 Käyttöliittymäkerros

Käyttöliittymäkerros on suunniteltu itsenäiseksi, mikä mahdollistaa kerroksen vaihtamisen helposti ohjelman päivitystilanteissa. Ohjelman osien vaihdettavuus saavutetaan siten, että kaikkia kerroksia ei tarvitse kääntää uudelleen muutettaessa ohjelman koodia yhdessä kerroksessa. Tämä on saavutettu suunnittelemalla rajapinta ja apuluokka käyttöliittymä- ja sovelluslogiikkakerrosten väliin. Kerroksessa sijaitsevat kaikki näkymät, joita ohjelmassa käytetään. Käyttöliittymäkerros suunniteltiin kommunikoimaan sovelluslogiikkakerroksen kanssa rajapinnan kautta. Käyttöliittymäkerroksen toimintoja kuvataan käyttötapauskaavioiden avulla. Käyttötapauskaaviot on luotu UML-standardin sääntöjä noudattaen ja Jude-järjestelmänsuunnittelutyökalua käyttäen. Määrittelyvaiheessa tehty käyttöliittymäkerroksen käsitetason luokkakaavio on liitteessä 2. Kuvassa 6 on ylimmän tason käyttötapauskaavio. Kaaviossa on kuvattu kaikki ohjelmassa suoritettavissa olevat toiminnot, jokainen käyttötapaus sisältää lisäkäyttötapauksia.



Kuva 6. Ylimmän tason toiminnot ohjelmassa.

Kuvassa 7 on kuvattu pääikkunan ylimmän tason toiminnat. Käyttötapauskaaviossa on kuvattu pääikkunan kautta suoritettavat toiminnot.



Kuva 7. Pääikkunan ylimmän tason käyttötapauskaavio.

4.4 Sovelluslogiikka-, tiedonsaanti- ja säilytyskerros

Sovelluslogiikka- ja tiedonsaantikerrosten sekä käyttöliittymäkerroksen luokat muodostavat moduulit, joihin kuuluvat näkymä, kontrolleri ja malli. Jokainen näkymä-kontrolleri-malliyhdistelmä muodostaa oman moduuliryhmän. Liitteessä 2 on esitetty sovelluslogiikka- ja tiedonsaantikerrosten käsitetason luokkakaaviot.

Sovelluslogiikkakerros on suunniteltu välittämään käyttöliittymäkerroksesta tulevat komennot alemmalle kerrokselle sekä alemmasta kerroksesta, tiedonsaantikerroksesta tulevat viestit ylemmälle kerrokselle. Kommunikointi ylemmän ja alemman kerroksen kanssa tapahtuu rajapinnan ja apuluokkien kautta. Sovelluslogiikkakerroksen komponentit eli kontrollerit on suunniteltu kommunikoimaan keskenään. UML-standardin ehdottaman säännön mukaan kommunikointi muiden näkymien ja mallien kanssa tulee tapahtua kontrollereiden kautta.

Tiedonsaantikerros on suunniteltu säilyttämään näkymien oletusarvot. Kerroksen tarkoitus on myös säilyttää ajonaikana syötettyjä arvoja. Kaikki mallit on sijoitettu tiedonsaantikerrokseen. Ainoastaan sarjaliikenneporttia, tiedostoa ja tietokantoja käsittelevien moduulien mallit ovat yhteydessä alemmalle säilytys- eli fyysiselle kerrokselle. Muissa moduuleissa alin saavutettava kerros on tiedonsaantikerros.

Säilytyskerroksessa sijaitsevat tietokannat, tiedostot ja sarjaliikenneportti. Säilytyskerros sijaitsee tietokoneen kiintolevyllä ja laitteistossa. Seuraavana kerrotaan tietokantojen suunnittelusta yleisesti.

Tietokantojen suunnittelu

Tietokantojen suunnittelussa on määritetty ensin vaatimukset tietokannoille. Vaatimukset on saatu keräämällä reitittimien komennoissa käytettyjä parametreja, kuten komennossa *interface FastEthernet0/0*. Komennolla asetetaan FastEthernet-portti asetusten tekemisen tilaan. Sen vaihtoehtoina ovat FastEthernet0/0 ja FastEthernet0/1 [21, s. 702–706]. Näistä on saatu selville eri tietokantojen ominaisuudet, eli mitä kenttiä niissä tulisi olla. Reititinkomentojen parametrien arvoista on saatu syöttöarvoja muodostettujen tietokantataulujen kentille. Taulukossa 2 on esimerkki muodostetusta reititinporttitaulusta.

Taulukko 2. Reititinporttien taulu.

Portintyyppi	Portintunnus
Ethernet	E 0/0
FastEthernet	Fa 0/0
Serial	s 0/0

Käyttäjän valitessa näkymässä haluamansa reititinportin ohjelma hakee vastaavan tiedon reititinporttitaulusta ja ilmoittaa näkymässä valittavissa olevat porttipaikat. Käyttäjä valitsee porttipaikan, ja ohjelma sallii kyseisen portin asetusten tekemisen.

5 Toteutus ja testaus

5.1 Tekninen toteutus

Ohjelman toteutus on suoritettu Microsoftin Visual Studio 2008 -kehitystyökalulla. Kehitystyökalun valintaan on vaikuttanut sen tarjoama uusi sarjaliikenneportin toiminnan hallintaan tarvittava luokkakirjasto System.IO.Ports. Luokkakirjasto sisältää sarjaliikenneportin hallitsemiseen tarvittavat ominaisuudet, kuten sarjaliikenneporttiresurssin, pariteetin ja lopetusbitin sekä sarjaliikenneportin avaamiseen ja sulkemiseen tarvittavat operaatiot. Tiedoston käsittelyyn on käytetty kehitystyökalusta saatavat tiedoston avaamis- ja tallentamisdialogit.

Ohjelman suunnittelu on suoritettu UML-standardia noudattaen, ja työvälineenä mallinnuksessa on käytetty Jude-järjestelmänsuunnittelutyökalua.

Ohjelman ohjelmointikielenä on käytetty C#-kieltä. Se on C-kieli, jossa on yhdistetty C++-kielen tehokkuus ja Java-kielen helppokäyttöisyys. C#-kieli on oliokieleksi muutettu C-kieli. Ohjelmoinnissa on noudatettu ohjelmointisääntöjä.

Tietokantojen taulut on tehty Microsoft Access 2003 -tietokantojen hallintaohjelmalla. Sen valitsin tietokantojen hallintaohjelmaksi, koska ohjelmaa tullaan käyttämään tietoverkkoluokkien työasemissa, joissa Microsoft Access 2003 on tuettu. Tietokantataulujen käsittelyyn ja tietokanayhteyden avaamiseen on käytetty Visual Studion tarjoamia luokkakirjastojen luokkia.

Yhteys tietokantaan avataan käyttämällä .NET Frameworkin tarjoamaa OleDb-luokkakirjastoa. Kirjasto sisältää tietokannan avaamiseen ja hallintaan tarvittavat seuraavat komennot:

OleDbConnection	Luokka, jossa on tietokantayhteyden tarjoaja ja tietokannan polku + nimi.

OleDbDataAdapter	Luokka jossa joukko komentoja ja tietokantayhteys, joita tarvitaan tietojoukon täyttämiseen.
OleDbCommand	Luokka, johon sisällytetään suoritettava SQL-komento. SQL-komento on tietokannan käsittelyyn omistettu komento.
OleDbCommandBuilder	Luokka, joka automaattisesti luo komentoja, joita käytetään sovittamaan tietojoukkoon tehdyt muutokset tietokannan kanssa.
DataSet	Luokka, jossa on tietojoukko-otos tietokannasta.
DataRow	Luokka, jossa on yhden tietokanta taulun rivi.

Prototyyppi

Prototyyppi on toteutettu yhtenä kokonaisena Visual Studio -projektina, josta on käännetty ajettava exe-tiedosto.

Tuoteversio 1.0

Tuote on toteutettu useina erillisinä Visual Studio -projekteina. Näkymistä koostuva käyttöliittymäkerrosprojekti on käännetty exe-tiedostoksi, jolla ohjelma käynnistetään käyttäjän toimesta. Sovelluslogiikka- ja tiedonsaantikerroksesta sekä rajapinnoista ja apuluokista on tehty omat projektit, ja ne on käännetty dll-tiedostoiksi.

5.2 Ohjelman testaus

Protoilumallin viimeisissä vaiheissa ohjelma testataan. Haikala ja Märijärvi (16, s. 40–41, 290) mainitsevat, että testauksen tarkoitus on löytää ohjelmistosta virheitä. Testaus suoritetaan yleensä monella tasolla kerroksittain. Testaustasoja ovat alimmalta tasolta ylemmäksi mentäessä moduuli-, integrointi- ja järjestelmätestaus. Moduuli- eli yksikkötestausvaiheessa testataan yksittäisen luokan operaatiot ja attribuutit. Seuraavalla tasolla suoritetaan integrointitestaus, jossa yhdistellään yhteen moduuleita tai moduuliryhmiä ja testataan moduulien välisten rajapintojen toimivuutta. Ylimmällä tasolla suoritetaan järjestelmätestaus. Siinä selvitetään koko ohjelmiston toiminta yhtenä kokonaisuutena sekä testataan ei-toiminnalliset ominaisuudet: kuormitustesti, luotettavuustesti, asennustesti ja käytettävyystesti.

Prototyyppi

Prototyyppin yksittäisiä osia tai moduuliryhmiä en testannut prototyypin kehitysvaiheen lopussa, vaan luotin kääntäjän toimintaan sekä suorittamaani järjestelmätestaukseen. Järjestelmätestauksessa testasin ohjelman toiminnan kokonaisuudessaan.

Tuoteversio 1.0

Tuotevaiheessa suoritin moduulitestauksen jokaiselle projektille eli kerrokselle. Testauksen suoritin Visual Studio 2008 -kehitystyökalun tarjoamalla testausprojektilla. Visual Studion testausprojektissa testattava projekti eli kerros liitetään testikohteeksi ja kehitystyökalu suorittaa tarvittavien testitapauksien kehittämisen ja itse testauksen. Järjestelmätestauksessa testasin järjestelmän toiminnallisuuden, ja samalla tuli testattua rajapintojen sekä apuluokkien oikeintoiminnallisuus.

6 Jatkokehitysmahdollisuudet

Graafinen konfigurointityökalu on kehitetty huomioiden jatkokehitysmahdollisuudet. Jatkokehitys on mahdollista muokkaamalla ohjelmakoodia, päivittämällä ohjelman käyttämiä tietokantatauluja, vaihtamalla kokonainen kerros toiseen tai lisäämällä kokonaisia näkymä-kontrolleri-mallimoduuliryhmiä.

Yhden luokan ohjelmakoodia muokkaamalla voidaan esimerkiksi muuttaa yhden näkymän toimintaa toisenlaiseksi tai vaihtaa koko näkymä uudentyyppiseksi. Ohjelma koostuu toisistaan irti olevista kerroksista, siten että yhden kerroksen voi vaihtaa, joutumatta kääntämään koko ohjelmaa uudelleen, kunhan kerros toteuttaa kerrostenväliset operaatiokutsut. Lisäämällä kokonaisia moduuliryhmiä voidaan ohjelmaan lisätä uusia toimintoja, kuten SSH-yhteyden avaamiseen ja hallintaan tarvittavia toimintoja. Ohjelmaa voi päivittää myös muuttamatta ohjelmakoodia. Tämä tapahtuu tietokantatauluja päivittämällä, esimerkiksi, kun ohjelman komennot halutaan asettaa tukemaan uudempaa reititintä, jossa reitittimen komennoissa on tapahtunut pieniä muutoksia.

7 Yhteenveto

Tässä insinöörityössä suunnittelin ja toteutin graafisen konfigurointityökalun, jota tullaan käyttämään apuna reititinlaitteiden perusasetusten tekemisessä. Ohjelmiston suunnittelussa en käyttänyt apuna vastaavanlaisia konfigurointityökaluja, mikä teki työn suunnittelusta ja toteutuksesta mielenkiintoisen, haastavan ja aikaa vievän. Työn tekeminen oli kuitenkin helpompaa, koska olin jo aikaisemmin perehtynyt verkkolaitteiden asetusten tekemiseen ja siten työn tavoite selvä. Työn yhteydessä opin työvaiheet ohjelmiston suunnittelusta aina sen toteutukseen asti ja sain paremman kuvan ohjelmistotuotannon vaiheista.

Ohjelman kehityksen yhteydessä kaikkia asiakasvaatimuksia ei täytetty. Alkuperäisen suunnitelman mukaan ohjelman tuli voida muuttaa monentyyppisten verkkolaitteiden asetuksia, mutta asiakasvaatimuksista poistettiin tuki monentyyppisille verkkolaitteille ja ohjelma muutettiin tukemaan vain reititintä. Siten esimerkiksi verkkopiirtoikkuna ei ollut enää tarpeellinen ominaisuus. Ongelmia esiintyi myös heikosti tehtyjen asiakasvaatimusten takia.

Lopullinen työ onnistui melko hyvin, vaikka muutama ongelma, joka ei johtunut itse ohjelmasta, jäi ratkaisematta.

Lähteet

1 Richter, Jeffrey, suom. Samela, Juha. Inside .NET Ohjelmointi. Helsinki: Edita Prima Oy, 2003.

2 Troelsen, Andrew. Pro C# 2008 and the .NET 3.5 Platform. Berkeley: Apress, Inc., 2007.

3 Demystifying Microsoft Intermediate Language. (WWW-dokumentti.) <http://www.devcity.net/Articles/54/msil_1_intro.aspx>. 2002. Luettu 26.3.2009.

4 Introducing Microsoft Intermediate Language. (WWW-dokumentti.) < http://www.dotnet-guide.com/msintermediate.html>. 2007. Luettu 26.3.2009.

5 Standard ECMA-335, Common Language Infrastructure (CLI). (WWW-dokumentti.) <http://www.ecma-international.org/publications/standards/Ecma-335.htm>. 2006. Luettu 27.3.2009.

6 Introducing the Common Language Runtime. (WWW-dokumentti.) < http://www.devhood.com/training_modules/dist-b/IntroCLR/introclr.htm>. 2001. Luettu 27.3.2009.

7 Technical Overview of the Common Language Runtime. (WWW-dokumentti.) < http://docs.msdnaa.net/ark/Webfiles/WhitePapers/CLR.pdf>. 2001. Luettu 27.3.2009.

8 .NET common language runtime (CLR) routines. (WWW-dokumentti.) <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.apdv.routines.doc/doc/c0011383.html>. 2008. Luettu 27.3.2009.

9 .NET Common Language Runtime Components. (WWW-dokumentti.) < http://www.informit.com/articles/article.aspx?p=30601 >. 2003. Luettu 27.3.2009.

10 NET framework Technologies. (WWW-dokumentti.) <http://msdn.microsoft.com/en-us/netframework/default.aspx>. 2009. Luettu 30.3.2009.

11 Koskimies, Kai. Ohjelmistoarkkitehtuurit. Jyväskylä: Gummerus Kirjapaino Oy, 2005.

12 Application Architecture Guide 2.0. (WWW-dokumentti.) <http://www.codeplex.com/AppArchGuide/Release/ProjectReleases.aspx?ReleaseId=20586>. 15.1.2009. Luettu 31.1.2009.

13 Bollapragada, Vijay. Inside Cisco IOS Software Architecture. Indianapolis: Cisco Press, 2000.

14 Allen, Tony. Internetworking Technologies Handbook. Indianapolis: Cisco Press, 2006.

15 Basic Software Configuration Using the Cisco IOS Command-Line Interface. (WWW-dokumentti.) <http://www.cisco.com/en/US/docs/routers/access/1800/1841/software/configuration/guide/b_cli.pdf>. 2004. Luettu 27.12.2008.

16 Haikala, Ilkka, Märijärvi, Jukka. Ohjelmistotuotanto. Jyväskylä: Gummerrus Kirjapaino Oy, 2006.

17 Puska, Matti. Yliopettaja, tietoverkot laboratorio. Metropolia Ammattikorkeakoulu, Espoon toimipiste, sähköpostikeskustelu 27.11.2008.

18 Puska, Matti. Yliopettaja, tietoverkot laboratorio. Metropolia Ammattikorkeakoulu, Espoon toimipiste, henkilökohtainen keskustelu 6.2.2009.

19 Rahkonen, Heikki. Projekti-insinööri, tietoverkot laboratorio. Metropolia Ammattikorkeakoulu, Espoon toimipiste, sähköpostikeskustelu 26.11.2008.

20 Koskimies, Kai. Pieni oliokirja. Jyväskylä: Gummerus Kirjapaino Oy, 1998.

21 Dooley, Kevin, Brown, Ian J. Cisco IOS Cookbook. Sebastopol: O'Reilly Media, Inc., 2006.

Liite 1: Vaatimustaulukko

Taulukko 1. Taulukossa on lueteltu vaatimukset ohjelmalle.

Vaatimus	Selitys	Toteutettu / Ei toteutettu
V1	Käyttäjän tulee voida lisätä verkkopiirtoikkunaan verkkolaitteita: Reitittimiä, kytkimiä ja työasemia, sekä yhdistellä verkkolaitteita kaapeleita vastaavilla kuvilla.	Ei toteutettu.
V2	Käyttäjän tulee voida siirrellä verkkolaitteita verkkopiirtoikkunassa.	Ei toteutettu.
V3	Käyttäjän tulee voida konfiguroida verkkolaitteita: reitittimiä ja kytkimiä, valitsemalla kyseinen verkkolaite verkkopiirtoikkunasta.	Reitittimien konfigurointi on toteutettu.
V4	Käyttäjän valittua verkkolaite verkkopiirtoikkunasta, ilmestyy käyttäjälle uusi konfigurointi -komentoikkuna, josta hänen tulee voida muokata verkkolaitteen asetuksia.	Ei toteutettu.
V5	Käyttäjän valittua komento tulee komennon asettua verkkolaitteelle, reaaliajassa, yhteyden salliessa niin.	Toteutettu.
V6	Käyttöliittymän tulee tulostaa tapahtumat verkkolaiteessa käyttäjän nähtäville, komentoikkunaan.	Toteutettu.
V7	Käyttäjän tulee voida suorittaa perusasetuksia verkkolaitteille: verkkolaitteiden nimet, salasanat, IP-osoitteet, reititysprotokollat, interface-, VLAN-, Frame Relay- ja VPN-asetukset.	Toteutettu. VLAN-asetus ja Frame Relay -konfigurointi on poistettu.

V8	Käyttäjän tulee voida valita SSH- tai telnet-yhteys, sekä normaaliyhteys, verkkolaitteeseen.	Toteutettu. SSH-yhteys on poistettu.
V9	Yhteys verkkolaitteelle tulee tapahtua COM-portin kautta Console-porttiin.	Toteutettu.
V10	Käyttäjän tulee voida valita COM-portti ja sen asetukset.	Toteutettu.
V11	Ilman käyttäjän valintaa COM-portin asetuksiksi tulee asettua oletusasetukset.	Toteutettu.
V12	Käyttäjän tulee voida asettaa yhteys päälle ja sammuttaa yhteys halutessa.	Toteutettu.
V13	Käyttäjän tulee voida tyhjentää verkkopiirtoikkuna.	Ei toteutettu.
V14	Käyttäjän tulee voida luoda uusi verkkopiirtoikkuna.	Ei toteutettu.
V15	Käyttäjän tulee voida tallentaa verkkopiirtoikkuna, jolloin kaikkien verkkolaitteiden asetukset tulee tallentua erillisiin tiedostoihin.	Ei toteutettu.
V16	Käyttäjän tulee voida tallentaa yksittäisen verkkolaitteen asetukset erilliseen tiedostoon.	Toteutettu.
V17	Käyttäjän tulee voida avata tallennettu asetus.	Toteutettu.
V18	Käyttäjän tulee voida ladata tekstimuotoinen konfiguraatio verkkolaitteeseen.	Toteutettu.
V19	Verkkolaitteet tulee esittää verkkopiirtoikkunassa aktiivi-ikoneina. Johtimet tulee esittää verkkolaitteina yhdistävänä viivoina.	Ei toteutettu.

V20	Käyttäjän tulee voida valita mitä verkkolaitteen ominaisuuksia verkkopiirtoikkunassa näkyy.	Ei toteutettu.
V21	Verkkopiirtoikkunassa tulee selvitä verkkolaitteelle konfiguroitu, käyttäjän laatima nimi.	Ei toteutettu.

Liite 2: Luokkakaavio