OULUN AMMATTIKORKEAKOULU

Antti Rajala

LISÄOMINAISUUDEN KEHITYS TOIMINNANOHJAUSJÄRJESTELMÄÄN

LISÄOMINAISUUDEN KEHITYS TOIMINNANOHJAUSJÄRJESTELMÄÄN

Antti Rajala
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Antti Rajala
Opinnäytetyön nimi: Lisäominaisuuden kehitys toiminnanohjausjärjestelmään
Opinnäytetyön englanninkielinen nimi: Development of an extra feature to ERP system
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Syksy 2021
Sivumäärä: 25

Opinnäytetyön toimeksiantajana oli Pinja Solutions Oy. Opinnäytetyössä luotiin uusi toiminto toimeksiantajan sahoille ja puunjalostuslaitoksille kehittämään toiminnanohjausjärjestelmään. Tarve toiminnolle tuli toimeksiantajan uuden asiakkaan myötä.

Opinnäytetyön teoriaosuus käsittelee .NET-arkkitehtuuria ja tarkemmin .NET Frameworkin toimintaa. Ohjelmisto, johon toiminto tehdään, on toteutettu C#- ja VB.NET-ohjelmointikielillä. Näiden kielien käyttäminen samassa ohjelmassa on mahdollista .NET Frameworkin tarjoamien ratkaisuiden avulla.

Uusi toiminto lähettää toimittajalle .csv-muotoisen tiedoston vastaanotetusta puukuormasta automaattisesti, kun kuorma on hyväksytysti vastaanotettu. Lähetys tapahtuu ohjelman taustalla häiritsemättä muuta ohjelman toimintaa. Tiedostojen asettelua voi muokata ja niiden lähetyksiä voi seurata ohjelmasta.

Asiasanat: C#, Visual Basic .Net, ohjelmointi, toiminnanohjausjärjestelmä

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Antti Rajala
Title of thesis: Development of an extra feature to ERP system
Supervisor: Kari Laitinen
Term and year when the thesis was submitted: Fall 2021
Number of pages: 25

This thesis was assigned by Pinja Solutions OY. In this thesis a new feature was created to an ERP system developed for sawmills and wood processing industry. Demand for this feature came from a new client of Pinja Solutions OY.

This thesis focuses on the theory of .NET architecture and more closely, the functionality of .NET Framework. The software, to which the new feature is added to, is done by using C# and VB.NET programming languages. The usage of these programming languages in the same software is made possible by the solutions offered by .NET Framework.

The feature automatically sends a file to a supplier in .csv format whenever a new wood load is received and acknowledged. Automatic sending of this file is done in the background of the software without disrupting other functionalities performed by the program. Layout and transmissions of the files can be monitored in the main program.

Keywords: C#, Visual Basic .Net, programming, ERP

SISÄLLYS

SANASTO

CIL/IL	Common Intermediate Language, jota Microsoft kutsuu välikieleksi. .NET-kielien kääntäjät kääntävät koodin välikieleksi, jonka JIT suorittaa
CLR	Common Language Runtime on ajonaikainen ympäristö, joka suorittaa välikielen yhdessä JIT:n kanssa
FTP	File Transfer Protocol eli tiedostonsiirtomenetelmä kahden tietokoneen välillä
JIT	Just In Time on kääntäjä, joka kääntää koodin konekielelle ohjelman suorituksen aikana
Keko	Keko on muistialue, jonka käyttämä tila muistista varataan ohjelman suorituksen aikana.
Kääntäjä	Kääntäjäksi kutsutaan ohjelmaa, joka kääntää korkean tason koodin konekielelle tai välikielelle
Pino	Pino on muistialue, jonka käyttämä tila muistista varataan ohjelman käännösvaiheessa

1 JOHDANTO

Opinnäytetyö tehtiin Pinja Solutions Oy:n metsäteollisuudelle kehittämään Timber by Pinja -ohjelmistoon. Timber by Pinja on sahojen ja puunjalostuslaitosten toiminnanohjausjärjestelmä.

Työn tavoitteena on tehdä ohjelmistoon toiminto, joka raportoi tukkitoimittajalle vastaanotetun puukuorman tiedot .csv-muotoisena tiedostona. Tiedosto pitää lähettää FTP:llä ja lähetyksen pitää tapahtua ohjelman taustalla automaattisesti aina, kun kuorman vastaanotto on valmistunut. Tiedostoja, joita käytetään, on tällä hetkellä noin 5 kappaletta ja näin ollen täytyy toteuttaa myös käyttöliittymään tapa hallita tiedostoja, jotta oikean muotoiset tiedostot lähtevät oikeaan paikkaan. Lähetetyistä tiedostoista tallennetaan tiedot tietokantaan, jotta niitä voi tarkastella myöhemmin ohjelmasta. Virheellisestä lähetyksestä on tultava ilmoitus käyttäjälle ja ohjelman toiminnan on jatkuttava normaalisti virheestä huolimatta. Myös tiedoston uudelleenlähetyksen on oltava mahdollista.

Ohjelmistossa on jo toiminto, jolla voi luoda Excel-tiedoston, johon on kerätty vastaanotetun kuorman tiedot. Tiedosto lähetetään tukkitoimittajalle aina hyväksytysti vastaanotetusta kuormasta. Tällä hetkellä tiedoston lähetys on suoritettava manuaalisesti.

Ohjelmointi toteutetaan Microsoftin Visual Studiolla. Opinnäytetyössä käydään läpi .NET-arkkitehtuurin toimintaa sekä ratkaisuja, joilla .NET on mahdollistanut usean eri ohjelmointikielen toiminnan samassa ohjelmassa. Koska tässä työssä kehitettävä toiminto on suoritettava ohjelman taustalla, käydään myös läpi, kuinka se toteutetaan ja minkälaista hyötyä se tarjoaa suoritettavalle ohjelmalle.

2 .NET-KEHITYSALUSTA

Ohjelmointiin tässä työssä käytettiin VB.NET- ja C#-kieliä. Kielien käyttämisen samassa ohjelmassa mahdollistaa Microsoftin kehittämä .NET-kehitysalusta, jota Microsoft Visual Studio -kehitysympäristö käyttää. Tämä työ tehtiin Windows-työpöytäsovellukseen, joten .NET:n tarjoamista ratkaisuista käyn tarkemmin läpi .NET Frameworkin toimintaa. On myös muita ratkaisuja, joista päällimmäisinä voisi mainita .NET Core, joka mahdollistaa .NET-ympäristössä kehitettyjen ohjelmien toiminnan myös macOS- ja Linux-käyttöjärjestelmissä. .NET Xamarin/Mono tarjoaa ratkaisut mobiiliapplikaatioiden rakentamiseen.

.NET on avoimen lähdekoodin alustariippumaton kehitysalusta ja soveltuu monenlaisten sovellusten kehitykseen. .NET:n avulla voidaan käyttää useita kieliä, editoreita ja kirjastoja web-, mobiili-, työpöytä-, peli- ja IoT-sovellusten rakentamiseen. (Microsoft 2021a.)

2.1 .NET Framework

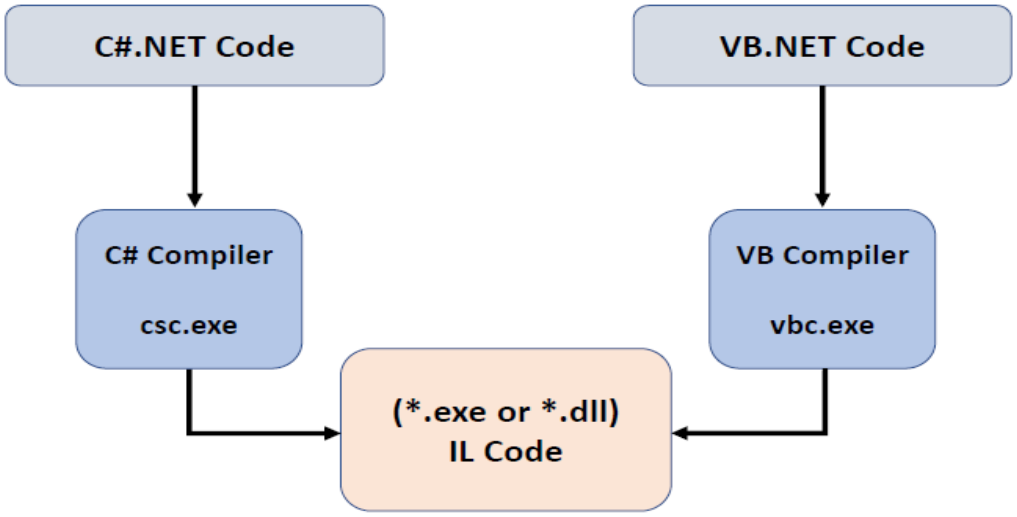
.NET Framework on ajonaikainen suoritusympäristö Windowsille, joka hallinnoi .NET -alustalle tehtyjä sovelluksia ja tarjoaa erilaisia palveluita käynnissä oleville sovelluksille. Framework koostuu kahdesta suuresta komponentista, ajonaikaisesta ympäristöstä CLR (engl. Common Language Runtime) ja luokkakirjastoista. CLR on moottori, joka hoitaa sovellusten suorituksen. Luokkakirjastot puolestaan tarjoavat sovelluskehittäjille testattua ja uudelleen käytettävää koodia. (Microsoft 2021b.)

2.2 Välikieli

Välikieli on kieli, jolle korkean tason lähdekoodi käännetään, ennen kuin se käännetään konekielelle. Microsoft kutsuu tätä välikieleksi, engl. Common Intermediate Language CIL tai IL, tekstissä tästä eteenpäin IL. Vielä ei ole kehitetty laitteistoa, joka suorittaisi IL:n suoraan ja siksi IL vaatiikin CLR:n koodin suoritukseen. Sovelluskehittäjien ei ole pakko ymmärtää täysin IL:n rakennetta, mutta voisi kuitenkin olla hyvä tietää hieman, mitä kääntäjät tekevät, kun suoritetaan koodia, joka on kirjoitettu .NET -kielillä tai muulla ajonaikaiseen suoritukseen perustuvalla kielellä. Myös Java-

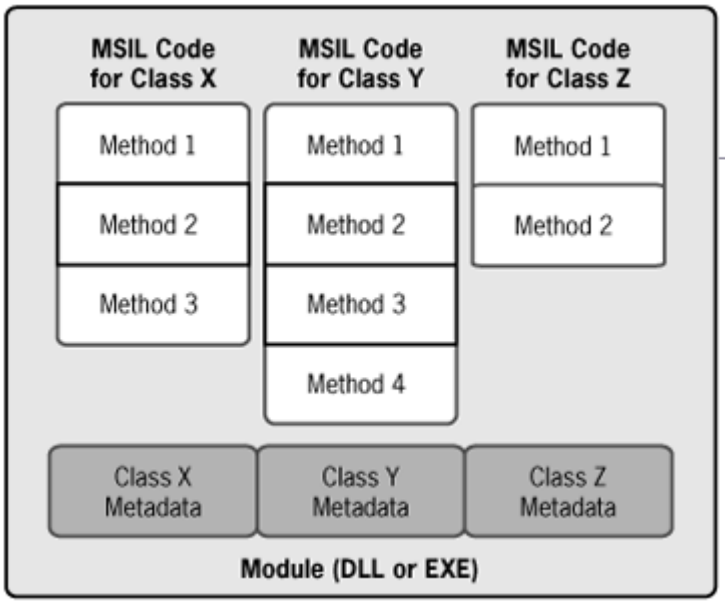
ohjelmointikielen kääntäjä kääntää koodin ensin välikielelle, jota kutsutaan tavukoodiksi, mutta Javan tavukoodi on suunniteltu vain tukemaan Javaa, kun IL tukee useampaa ohjelmointikieltä. (Chappell 2006, luku 2.)

Kuvassa 1 havainnollistetaan kielten omien kääntäjien toimintaa ja IL-tiedoston muodostumista. Kääntäjien luomaa tiedostoa kutsutaan myös moduuliksi.



KUVA 1. IL-tiedoston muodostus (Posadas 2016, luku 1)

Kun koodi on käännetty IL:ksi, tulee samassa tiedostossa aina mukana myös metatiedot luokista, jotka tiedosto sisältää. Metatiedot pitävät sisällään luokkien nimet, näkyvyyden, tiedon luokan periytyvyydestä, rajapinnat, joita luokka toteuttaa, sekä metodit, muuttujat ja palvelut. Metodien sisältämät muuttujat, tyypit ja mahdolliset palautukset ovat myös kuvattuna metatiedoissa (kuva 2). (Chappell 2006, luku 2.)



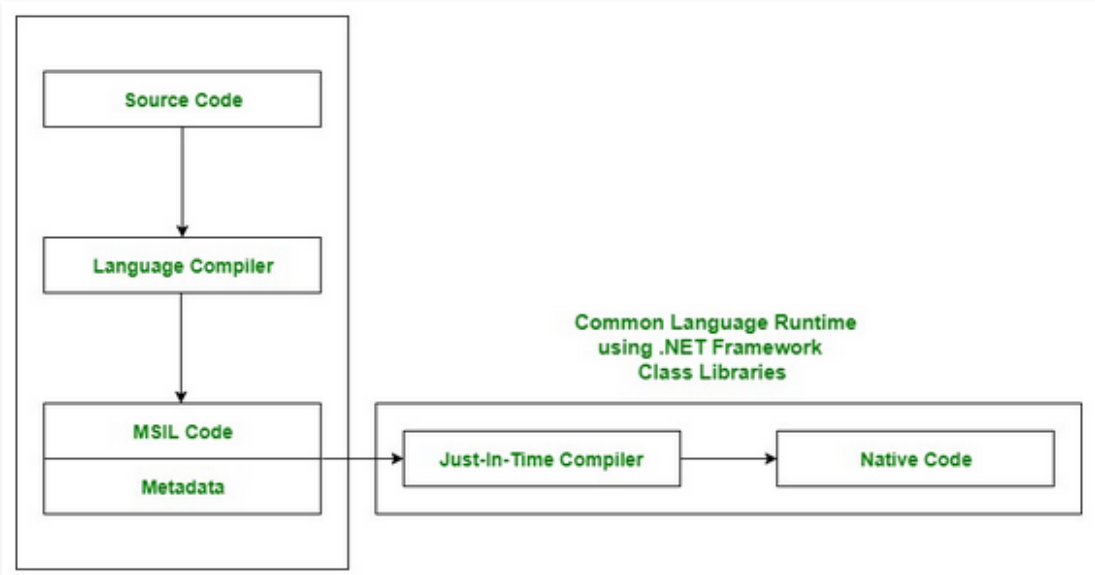
KUVA 2. IL-tiedoston rakenne (Chappell 2006, luku 2)

Kääntäjät ja CLR noudattavat samaa julkista standardia, ECMA Common Language Infrastructure. ECMA:ssa on määritelty metatietojen muoto, tyyppijärjestelmä ja virtuaalinen suoritusjärjestelmä. (Microsoft 2021c.)

2.3 .NET CLR

CLR eli Common Language Runtime on alustariippumaton ajonaikainen ympäristö, joka on perustana kaikelle .NET Frameworkin toiminnalle. Jos haluaa ymmärtää .NET-kieliä tai .NET-luokkakirjastoja, on ymmärrettävä CLR. Koodia, joka käännetään CLR:n suoritettavaksi, kutsutaan hallituksi koodiksi. CLR tarjoaa monipuoliset työkalut, jotka tukevat koodin kirjoitusta ja suoritusta. (Chappell 2006, luku 2.)

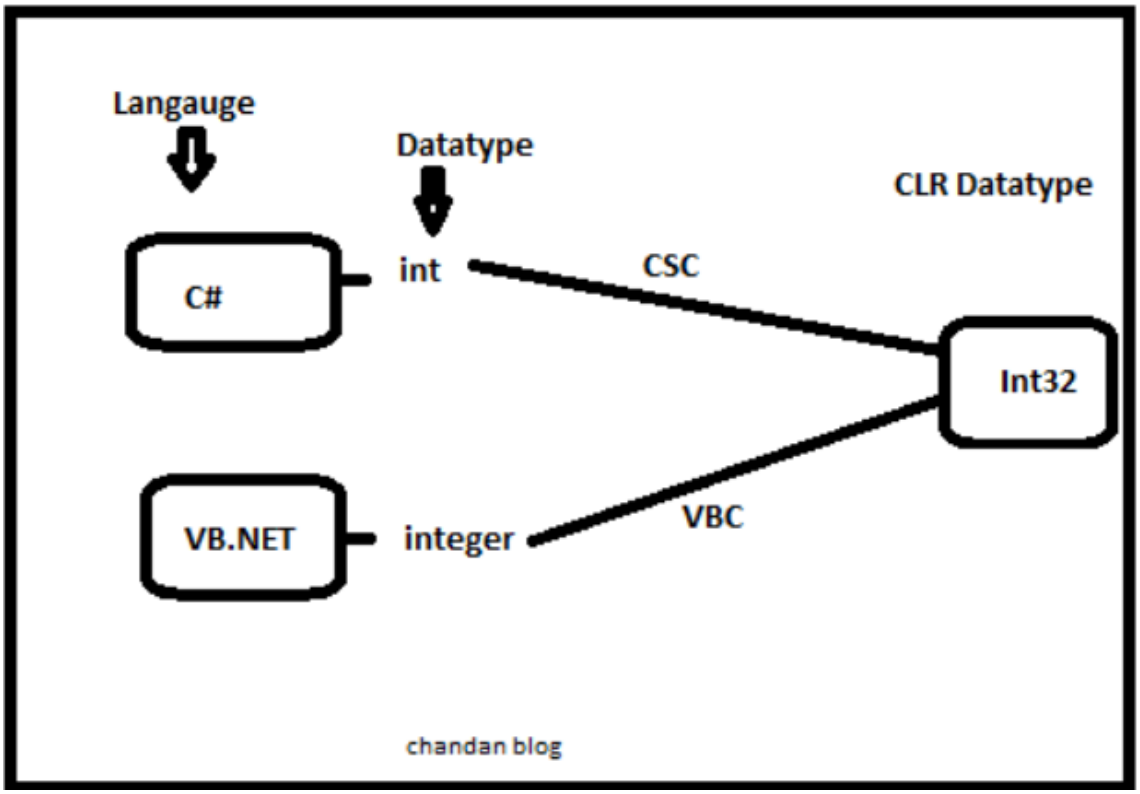
CLR parantaa ohjelmien suorituskykyä ja siirreltävyyttä eri käyttöjärjestelmien välillä ilman, että koodia tarvitsisi uudelleen kääntää. CLR analysoi ja tarkistaa koodikielen oikeellisuuden, ennen kuin se annetaan prosessorin suoritettavaksi, joten sovelluskehittäjän ei tarvitse itse miettiä, kääntyykö koodi konekielelle oikein. Yleinen tyyppijärjestelmä CLR:n sisällä mahdollistaa kirjastojen ja komponenttien käytön eri ohjelmointikielien välillä. Kuvassa 3 on prosessi lähdekoodista konekielelle. (GeeksforGeeks 2021.)



KUVA 3. Prosessi lähdekoodista konekielelle (GeeksforGeeks 2021)

2.3.1 Common Language Specification

Common Language Specification eli CLS on yleinen kielimääritys, jonka tehtävänä on konvertoida eri .NET-ohjelmointikielien syntaksi muotoon, jota CLR ymmärtää. Kuten kuvassa 4 on kuvattu, konvertoituvat eri ohjelmointikielillä esitetyt saman datatyypin muuttujat käännösvaiheessa yhteisen tyyppijärjestelmän määrittämäksi datatyypiksi (Kumar 2019). CLS onkin yksi niistä tekijöistä, jotka mahdollistavat eri ohjelmointikielien käytön samassa ohjelmassa. (GeeksforGeeks 2021.)

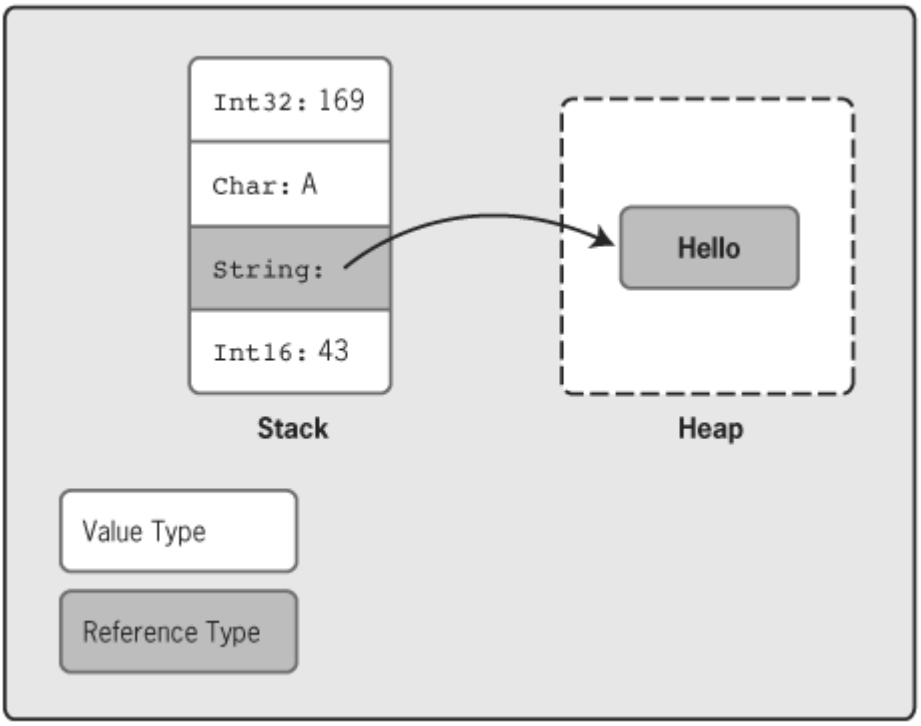


KUVA 4. Yhteisen kielen määritys (Kumar 2019)

2.3.2 Common Type System

Common Type System eli CTS on vastuussa siitä, että jokaisen .NET:n tarjoamien ohjelmointikielien tyyppijärjestelmät konvertoituvat sellaiseksi tyyppijärjestelmäksi, jota CLR ymmärtää, eli yleiseen muotoon. Jokaisella ohjelmointikielellä on oma tyyppijärjestelmä. (GeeksforGeeks 2021.)

On olemassa kaksi CTS-datatyyppiä määriteltynä niiden käyttämän muistin mukaan, jotka kaikilla .NET-kielillä on, arvotyypit ja viittaustyypit. Arvotyypit ovat pinoon (engl. stack) sijoittuvia datatyyppejä, jotka on alustettu esim. merkkijonoksi tai kokonaisluvuksi. Arvotyyppien käyttämä muisti vapautetaan automaattisesti esimerkiksi, kun metodin sisällä luodaan muuttuja ja metodin suoritus loppuu. Viittaustyypit ovat datatyyppejä, joiden alustettu datatyyppi on pinossa, mutta niiden sisältämä arvo löytyy keosta (engl. heap). Tällaisia voivat olla esimerkiksi luokasta luodut oliot sekä listat, jotka sisältävät saman tyyppisiä arvoja. Viittaustyyppien käyttämä muisti vapautetaan .NET-ohjelmissa suorituksen aikana roskien keruun yhteydessä. Kuvassa 4 havainnollistetaan datatyyppien sijoittuminen eri muistialueille. (Chappell 2006, luku 2.)



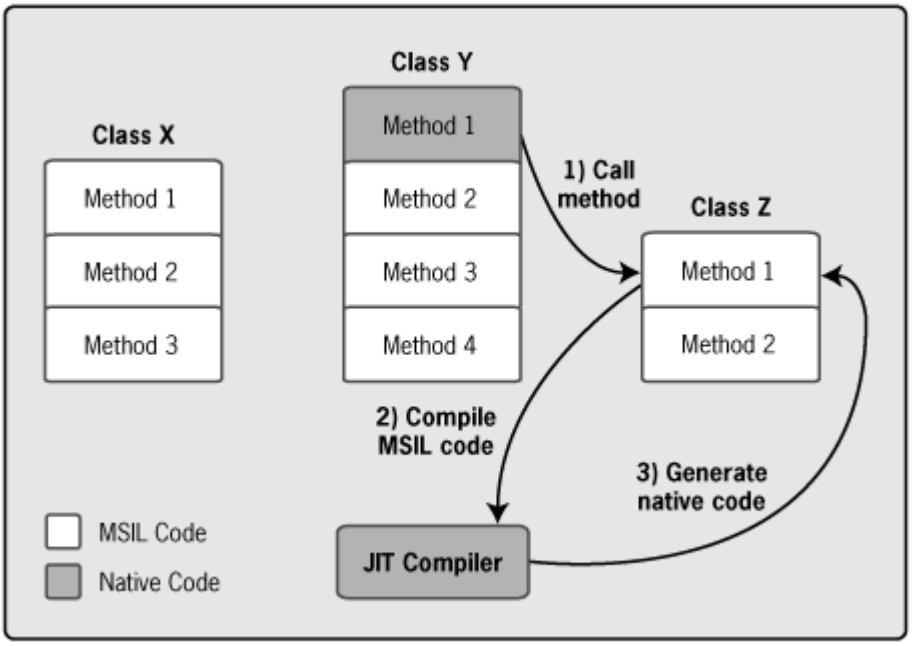
KUVA 5. Datatyyppien sijoittuminen muistialueille (Chappell 2006, luku 2)

2.3.3 Garbage collector

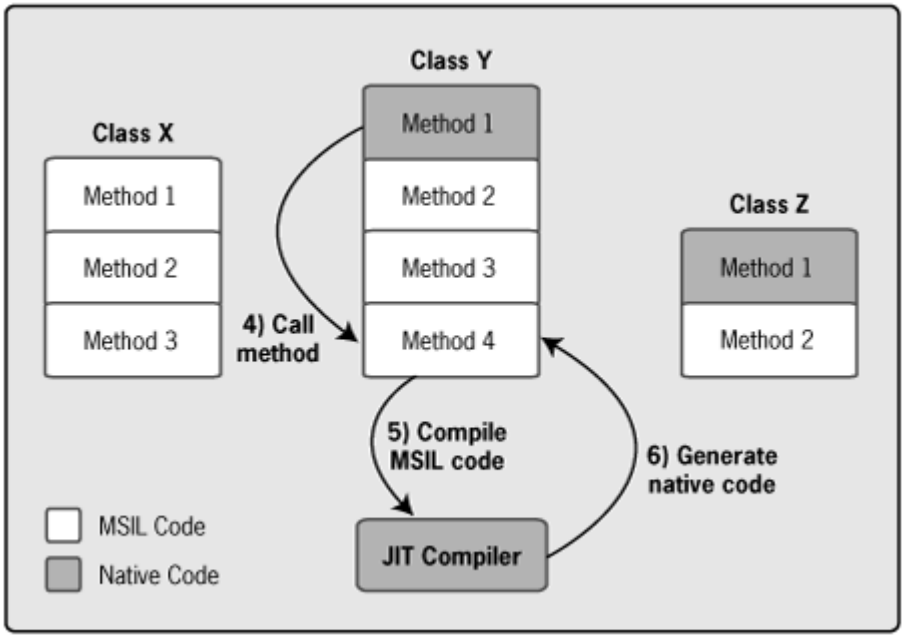
Garbage collector eli roskien keruu on automaattista muistinhallintaa. Muistinhallinta on tärkeässä roolissa, kun CLR huomaa, että keko-muistialue alkaa täyttyä ohjelman suorituksen aikana. Roskaksi keossa voidaan ajatella esimerkiksi arvoa, johon ei enää viitata pinon puolelta. Viittauksen puuttuessa arvo voidaan poistaa ja sen käyttämä muisti vapauttaa. (Chappell 2006, luku 2.)

2.3.4 JIT Compiler

Yleisin tapa kääntää IL on antaa CLR:n ladata kaikki metatiedoissa kuvatut tiedostot ja kääntää IL:n sisällä oleva koodi konekielelle vasta, kun ohjelma sitä tarvitsee. Tätä prosessia kutsutaan ajonaikaiseksi kääntämiseksi (engl. Just In Time compiler). Kuvissa 5 ja 6 on kuvattu tätä prosessia. Kun IL-koodia on kerran käännetty konekielelle, ei JIT-kääntäjä uudelleen käännä sitä, vaan se pysyy muistissa ohjelman suorituksen ajan. (Chappell 2006, luku 2.)



KUVA 6. JIT-käännös kutsuttaessa Z-luokan metodia (Chappell 2006, luku 2)



KUVA 7. JIT-käännös kutsuttaessa Y-luokan metodia (Chappell 2006, luku 2)

3 ASYNKRONINEN OHJELMOINTI

Asynkroninen ohjelmointi on tapa saada ohjelmasta paljon käyttäjäystävällisempi. Ohjelman toiminnot ovat käytössä koko ajan ilman, että tarvitsee odottaa edellisen toiminnan suorittamista loppuun. Tiedoston automaattisen lähetyksen on suoriuduttava ohjelman taustalla, joten tämä luku sisältää tietoa säikeistä ja prosesseista.

Useimmiten koodi on kirjoitettu synkroniseen muotoon, jolloin ohjelma suorittaa koodin siinä järjestyksessä, mihin se on kirjoitettu. Hyvin rakennettu ja jäsennelty synkroninen koodi on helposti ylläpidettävä, verrattuna asynkroniseen suoritukseen. Kuitenkin jos synkronisessa koodissa tulee toiminto, jonka suoritukseen kuluu paljon aikaa, pitää toiminnon suoritus odottaa loppuun, ennen kuin ohjelman käyttöä voi jatkaa. Pitkään kestävän toiminnon siirtäminen omaan säikeiseen korjaa tämän. Asynkroninen toiminta voidaan toteuttaa joko usealla eri koneella, usealla prosessilla tai jakamalla suoritus säikeisiin. (Blewett & Clymer 2013, luku 1.)

3.1 Prosessi

Prosessi on jokin suoritettava ohjelma ja sen suoritukseen käyttöjärjestelmä käyttää prosessielementtiä (Process Control Block eli PCB). Jokaiselle prosessille on oma prosessielementti käyttöjärjestelmän ytimessä, johon käyttöjärjestelmä tallentaa jokaisen prosessin tiedot. Tietoja, joita elementti sisältää suoritettavasta prosessista, ovat esimerkiksi prosessin paikka pinossa, prosessin nimi ja prosessin tila. Käyttöjärjestelmä käyttää näitä tietoja prosessien suorituksen hallintaan. (GreeksforGreeks 2020.)

Prosessien käytössä oleva muisti on varattuna vain prosessille itselleen eikä se voi suoraan käyttää muiden prosessien muistissa olevia resursseja. Tämä mahdollistaa sen, että jos jonkin muun ohjelman suoritus jumiutuu ja se joudutaan sammuttamaan, sillä ei ole vaikutusta muiden ohjelmien suoritukseen (Bauer 2017). Prosessi koostuu yhdestä tai useammasta säikeestä (Microsoft 2021d).

3.2 Säie

Säie on prosessin osa, jolle käyttöjärjestelmä allokoi suoritusaikaa prosessorilta (Microsoft 2021d). Säikeillä on käytössä prosessin käyttämä muisti ja resurssit, mutta ne eivät voi prosessin lailla vaikuttaa muihin prosesseihin. Muihin säikeisiin saman prosessin alla ne voivat kyllä vaikuttaa. Säikeille varataan oma muistipaikka pinosta (stack), joka on käytössä vain sillä säikeellä, jolle se varataan, mutta keossa (heap) oleviin tietoihin pääsevät käsiksi kaikki prosessin säikeet. Windows ohjaa säikeiden suoritusta omalla komponentilla, Thread Scheduler. Thread Scheduler antaa säikeille suoritusaikaa prosessorilta sen mukaan, mikä prioriteetti on prosessilla ja säikeellä. (Blewett & Clymer 2013, luku 1.)

3.3 Task

.NET tarjoaa ohjelmointirajapinnan, Task Paraller Library eli TPL, jolla voi toteuttaa asynkronisia operaatioita suoritettavaan ohjelmaan. .NET 4.0 -versiossa esiteltyä TPL-rajapintaa onkin suositeltu käytettäväksi asynkronisten ja monisäikeisten toimintojen toteuttamiseen (Microsoft 2021e). Aikaisemmat .NET-rajapinnat asynkronisiin operaatioihin olivat hankalasti ymmärrettäviä ja vaikeakäyttöisiä, sekä rajapinnoista puuttui toiminnot, joilla suoritettavaa tehtävää olisi voinut seurata tai lopettaa (Cook 2019).

TPL mahdollistaa asynkronisten toimintojen hallinnan tasolla, mikä sovelluskehittäjien olisi vaikea itse toteuttaa (Cook 2019). Näitä toimintoja ovat esimerkiksi tehtävien jakaminen säikeille, säikeiden oikea-aikainen suoritus Threadpoolista, säikeiden peruuttaminen ja tilan vaihtaminen sekä muita matalan tason toimintoja. Sovelluskehittäjä pystyy TPL:n avulla keskittymään itse toiminnallisuuden ohjelmointiin, ilman että hänen täytyy puuttua edellä mainittuihin toimintoihin ja samalla maksimoimaan ohjelman suorituskyvyn. (Microsoft 2021e.)

TPL pohjautuu Task-luokkaan, jolla asynkroniset toiminnot tehdään. Task-objektit ovat toisistaan riippumattomia itsenäisiä operaatioita. Taskit kerätään työjonoon, josta TPL:n Task Scheduler ne poistaa ja suorittaa, yleensä silloin, kun prosessorilta vapautuu ydin sen suorittamiseen. Toisin kuin luotaessa uusi säie, task-objektia ei suoriteta heti. (Microsoft 2010.)

TPL:n käyttö vaatii prosessorilta suoritusaikaa ja monimutkaistaa ohjelman suorittamista, minkä vuoksi kaikkea koodia ei ole järkevää laittaa TPL Taskin suoritettavaksi. Esimerkiksi toiminto, jonka

suoritukseen ei mene prosessorilta kauan aikaa, on hyvä jättää pääsäikeeseen suoritettavaksi. (Microsoft 2021e.)

4 TOIMINNON TOTEUTUS

Tarve uudelle toiminnolle tuli Pinja Solutions Oy:n uuden asiakkaan myötä. Asiakas raportoi toimittajalle vastaanotetun puukuorman tietoja .csv-muotoisella tiedostolla. Tiedosto lähetetään FTP-protokollaa käyttäen. Aikaisemmin tiedosto luotiin ja lähetettiin manuaalisesti ohjelmasta. Tähän asiakas halusi muutoksen, jolla tiedoston luonti ja lähetys tapahtuisi automaattisesti, kun kuorma on hyväksytysti vastaanotettu. Automatisointi luonnollisesti vähentäisi lajittelijan työkuormaa.

4.1 Tekniikat

Kehitysalustana toimi Visual Studio ja ohjelmointikielinä VB.Net ja C#. Tietokannat on toteutettu SQL Serveriä käyttäen ja niiden hallinta tapahtui SQL Server Management Studiolla (SSMS).

4.2 Suunnittelu

Ennen varsinaisen ohjelmoinnin aloitusta pidettiin aloituspalaveri, jossa käytiin läpi vaatimukset, jotka toiminnolle oli asetettu. Palaverissa sovittiin myös, että käytössä olevista raporteista ja uuden asiakkaan raporteista tehtäisiin ennen ohjelmoinnin aloitusta Excel-tiedostoon vertailumatriisi. Matriisista näkisi raporttien eroavaisuudet ja pystyisi päättämään ylläpitoa ajatellen, mitkä tiedot pitäisi pystyä piilottamaan ja missä tilavuusyksikössä tilavuudet näkyisivät. Uusista tietokantatauluista ja toimintoa ohjaavista luokista luotiin vertailumatriisin valmistumisen jälkeen alustava suunnitelma. Toteutuksen suhteen minulla oli vapaus päättää, kuinka tämän työn toteutan. Työn etenemistä seurattiin ketterille ohjelmistokehitysmenetelmille tyypillisillä viikoittaisilla palavereilla.

4.3 Toteutus

4.3.1 Tietokanta

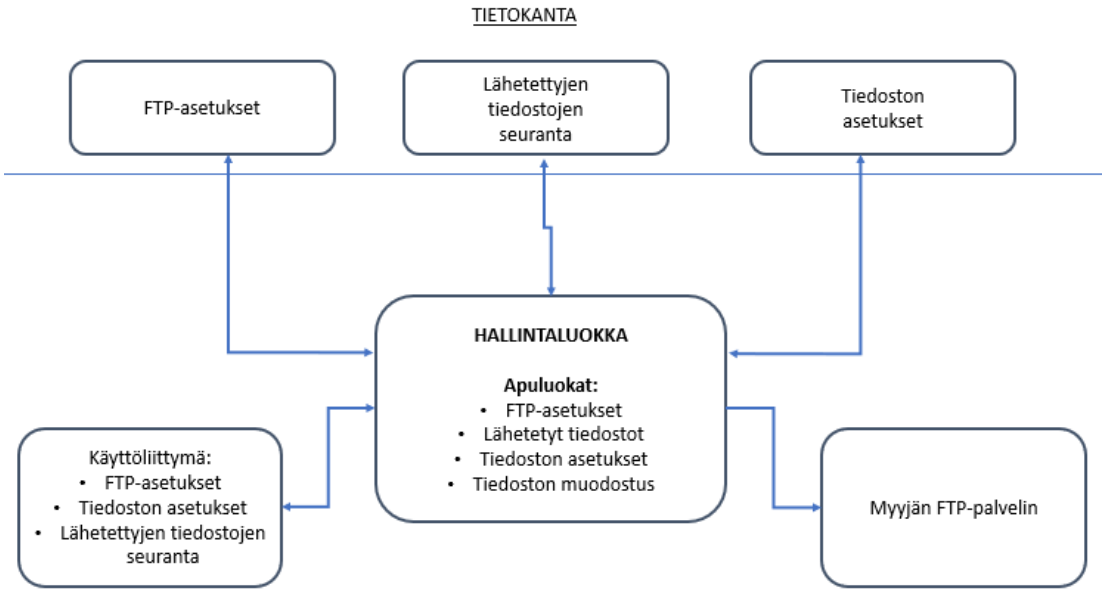
Tietokantaan lisättiin kolme uutta taulua. Ensimmäinen taulu pitää sisällään toimittajalle asetetut asetukset, joita ovat FTP-tiedot, automaattisen lähetyksen tila ja raportille valittu pohja (valmis tai

muokattu). Toista taulua käytetään lähetettyjen tiedostojen seurantaan. Taulusta saadaan toimittajan tiedot, jolle tiedosto on lähetetty, sekä kuormanumero, jonka perusteella tiedosto on kasattu. Lähetetyn tiedoston sisältö ja mahdollinen virheviesti ongelmatilanteessa käy myös ilmi toisesta taulusta. Kolmanteen tauluun tallennetaan muokatun tiedostopohjan asetukset. Nämä asetukset eivät ole toimittajakohtaisia ja ovat valittavissa usealle toimittajalle.

4.3.2 Luokat

Luokkarakenteesta en tehnyt kovin monimutkaista. Yksi hallintaluokka toimii ”moottorina” ja hoitaa tiedoston muodostuksen, lähetyksen ja tietokannan päivityksen. Tietokantatauluista on luotu apuluokkia, jotka toimivat tietovarantona. Näitä apuluokkia käytetään tietojen esittämiseen käyttöliittymässä. Käyttäjän muokatessa tietoja käyttöliittymässä voidaan apuluokka antaa parametrinä hallintaluokan metodille, joka päivittää muuttuneet tiedot tietokantaan. Tietokantatauluista luodut apuluokat eivät sisällä lainkaan metodeja.

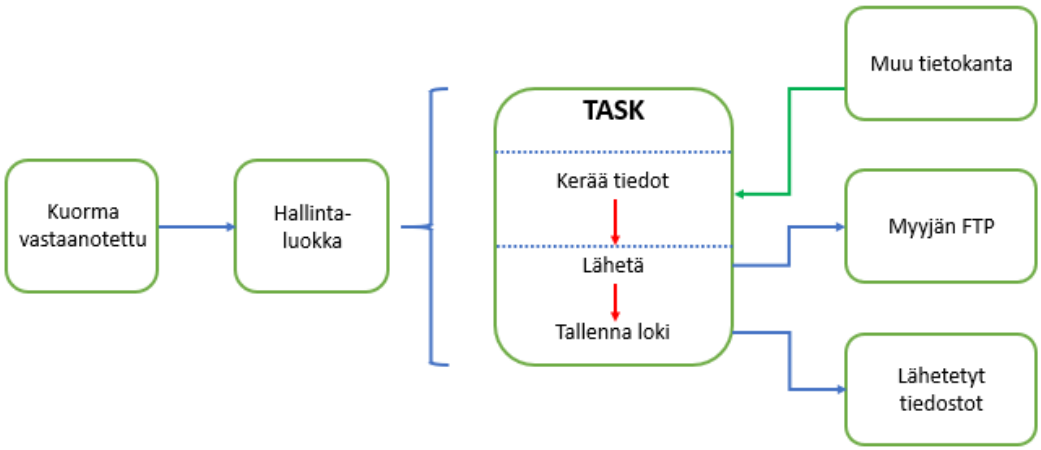
Tiedoston muodostukseen käytin tekemääni abstraktia tiedostoluokkaa, jonka aliluokat on nimetty valmiiden tiedostopohjien mukaan. Nämä aliluokat toteuttavat tiedostoluokassa esitetyn tiedoston muodostusmetodin, joka palauttaa tiedoston sen mukaan, mistä aliluokasta ilmentymä on luotu. Tiedoston muodostusmetodi ottaa parametrinä tukkidataolion, johon on kerätty tietokannasta kaikki tiedoston tarvitsemat tiedot. Parametrin saatuaan metodi purkaa tukkidataolion sisältämät arvot lähetettävään tiedostoon omalle paikalleen. Myös muokatulle pohjalle on oma aliluokka ja käytössä sama metodi, mutta erona edellisiin on käyttäjän määrittämät tiedostoasetukset tallennettuna luokan muuttujalle, joiden mukaan arvot sitten tiedostoon asetetaan. Kuvassa 8 on toteutetun toiminnon keskeisimmät asiat, pois lukien tukkidatan muodostukseen käytettävät taulut ja luokat.



KUVA 8. Toteutettavan toiminnon sisältö

4.3.3 Lähetys

Kuorman vastaanoton valmistuessa ohjelma tarkistaa toimittajan asetuksista automaattisen tiedoston lähetyksen tilan ja lähetykseen vaaditut asetukset. Kun tarkistukset on tehty, luodaan hallintaluokasta olio, joka pitää sisällään kaikki tarvittavat asetukset. Tämä luokka välitetään parametrinä kaksiosaiselle Taskille. Taskin ensimmäinen osa kerää tarvittavat tiedot tietokannasta. Toinen osa muodostaa tiedoston ja lähettää sen. Lähetyksestä tallentuu myös merkintä tietokantaan seurantaa varten. Taskin käyttäminen useammassa osassa on mahdollista luokan tarjoaman ContinueWith-metodin avulla, joka pakottaa odottamaan edellisen Taskin suorituksen loppuun, ennen kuin seuraavan voi aloittaa. Seuraavalla aloitettavalla Taskilla on käytössä edellisen Taskin tulokset, jotka tässä tapauksessa ovat tukkidatat vastaanotetusta kuormasta. Virheen sattuessa palautuu käyttäjälle viesti, että jokin meni pieleen, muun ohjelman suorituksen jatkuessa normaalisti. Kuvassa 9 on lähetyksen eri vaiheita.



KUVA 9. Lähetyksen vaiheita

Kuormien vastaanotto on ohjelmoitu VB.Net -kielellä ja tiedoston lähetys ohjelman taustalla ohjelmoitiin myös samalla kielellä. Jotta virheviestin sai näkymään käyttäjälle Taskin sisältä, oli siihen luokkaan, jonka sisälle lähetys ohjelmoitiin, luotava julkinen jaettu tapahtuma, jota käyttöliittymä säikeeseen asetettu tapahtumankäsittelijä kuuntelee. Jos jostain syystä tiedoston muodostuksessa tai lähetyksessä tapahtuu virhe, Taskin virheenkäsittelyn sisältä välitetään virheviesti käyttöliittymälle luokalle määritetyn jaetun tapahtuman avulla.

4.3.4 Käyttöliittymä

Toimittajien ylläpitoon lisättiin kaksi välilehteä, asetukset ja tapahtumien seuranta. Asetukset-välilehdellä on mahdollista asettaa toimittajan FTP-asetukset ja tiedoston automaattisen lähetyksen tila. Samalla välilehdellä päätetään myös valmiin tai muokatun tiedosto pohjan käyttö. Jos käyttöön halutaan muokattu pohja, pystyy näkymästä valitsemaan määräarvoille tilavuusyksikön ja desimaalien lukumäärän, joilla määräarvot esitetään. Myös tietojen piilotus tiedostolta on mahdollista muokatulle pohjalle. Asetukset tallentuvat valittuna olevalle toimittajalle ja samat tiedostoasetukset voi valita myös toiselle toimittajalle.

Tapahtumien seurannasta oletuksena näkee valittuna olevan toimittajan lähetykset. Tapahtumien seuranta -välilehdellä on myös mahdollista aikarajata lähetysten esitystä taulukossa sekä valita kaikki ne toimittajat, joiden lähetyksiä halutaan tarkastella. Tiedostoista, joiden lähetyksessä on

tapahtunut virhe, tallentuu seurantaan merkintä, jossa on mukana virheviesti tapahtuneesta virheestä. Kuormien vastaanotto -näkymästä on mahdollista lähettää kuorman tiedot uudelleen toimittajalle.

5 YHTEENVETO

Opinnäytetyön aiheena oli lisätä sahoille ja puunjalostuslaitoksille kehitettyyn toiminnanohjausjärjestelmään toiminto, jolla voi raportoida vastaanotetun kuorman tiedot .csv muotoisena tiedostona toimittajalle. Tiedoston lähetys on suoritettava ohjelman taustalla, kun kuorma on hyväksytysti vastaanotettu. Työhön käytettiin VB.NET- ja C# -ohjelmointikieliä ja niiden käyttämisen yhdessä mahdollisti Microsoftin kehittämä .NET -kehitysalusta.

Toiminnanohjausjärjestelmään tehtiin tukkimyyjien ylläpidon yhteyteen Asetukset-välilehden FTP- / tiedostoasetuksille ja Tapahtumien seuranta -välilehden lähetettyjen tiedostojen seurannalle. Asetukset-välilehdellä voi FTP-asetusten lisäksi määrittää tietojen asettelun tiedostoon. Asetteluille on käytössä valmiita pohjia ja lisäksi omien pohjien lisääminen on mahdollista. Lisätyt pohjat ovat käytössä myös muille toimittajille. Tapahtumien seuranta -välilehdelle tallentuvat lähetettyjen tiedostojen lisäksi myös ne tiedostot, joiden lähetyksessä on tapahtunut virhe. Lisäksi tiedoston manuaalinen lähetys on mahdollista ohjelmasta. Tehdystä toiminnosta ei liikesalaisuuksien vuoksi voi kirjoittaa tarkempaa kuvausta.

Pääsin toiminnolle asetettuihin vaatimuksiin annetussa ajassa ja työ voitiin toimittaa asiakkaalle. Toiminnon kehitystyö jatkuu asiakkaalta saadun palautteen perusteella.

Käytettävissä olleet tekniikat soveltuivat hyvin toiminnon toteuttamiseen. Työtä tehdessä ohjelmointitaidot ja tietokantojen käyttötaidot kehittyivät. Tietokantahakujen optimointi indeksien avulla tuli hyvänä lisänä omaan osaamiseen.

LÄHTEET

Bauer, Roderick 2017. What's the Diff: Programs, Processes, and Threads. Backblaze. Hakupäivä 6.9.2021. https://www.backblaze.com/blog/whats-the-diff-programs-processes-and-threads/.

Blewett, Richard & Clymer, Andrew 2013. Pro Asynchronous Programming with .NET. Apress. Hakupäivä 20.9.2021. O´Reilly. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/pro-asynchronous-programming/9781430259206/.

Chappell, David 2006. Understanding .NET, Second Edition. Addison-Wesley Professional. Hakupäivä 5.9.2021. O´Reilly. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/understanding-net-second/0321194047/.

Cook, Nate 2019. Why the Task Parallel Library Should Matter to You. Async Programming With the Task Parallel Library. Pluralsight. Hakupäivä 2.9.2021. https://www.pluralsight.com/guides/async-programming-task-parallel-library.

GeeksforGeeks 2021. Common Language Runtime (CLR) in C#. Hakupäivä 5.9.2021. https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/.

GreeksforGreeks 2020. Process Table and Process Control Block (PCB). Hakupäivä 6.9.2021. https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/.

Kumar, Chandan 2019. What Are CTS And CLS In .NET. C# Corner. Hakupäivä 6.9.2021. https://www.c-sharpcorner.com/blogs/what-are-cts-and-cls-in-net.

Microsoft 2010. Parallel Tasks. Parallel Programming with Microsoft .NET. Hakupäivä 6.9.2021. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff963549(v=pandp.10)#tasks-and-threads.

Microsoft 2021a. Introduction to .NET. .NET documentation. Hakupäivä 5.9.2021. https://docs.microsoft.com/fi-fi/dotnet/core/introduction.

Microsoft 2021b. Get started with .NET Framework. .NET Framework documentation. Hakupäivä 3.9.2021. https://docs.microsoft.com/fi-fi/dotnet/framework/get-started/?WT.mc_id=dotnet-35129-website#Introducing.

Microsoft 2021c. Common Language Runtime (CLR) overview. .NET documentation. Hakupäivä 3.9.2021. https://docs.microsoft.com/en-us/dotnet/standard/clr.

Microsoft 2021d. Processes and Threads. Hakupäivä 7.9.2021. https://docs.microsoft.com/en-us/windows/win32/procthread/processes-and-threads.

Microsoft 2021e. Task Parallel Library (TPL). Hakupäivä 6.9.2021. https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl.

Posadas, Marino 2016. Mastering C# and .NET Framework. Packt Publishing. Hakupäivä 6.9.2021. O´Reilly. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/mastering-c-and/9781785884375/.