jamk.fi

Firstbeat-analytiikkatyökalu

Jani Kähkönen

Opinnäytetyö
Joulukuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), Ohjelmistotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kähkönen, Jani	Opinnäytetyö, AMK	Joulukuu 2018
	Sivumäärä 38	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Firstbeat-analytiikkatyökalu

Tutkinto-ohjelma
Ohjelmistotekniikan tutkinto-ohjelma

Työn ohjaaja(t)
Rantala Ari, Huotari Jouni

Toimeksiantaja(t)
Firstbeat Technologies Oy

Tiivistelmä

Firstbeat Technologies Oy on vuonna 2002 Jyväskylässä perustettu hyvinvointialan ohjelmistoyritys, joka on kehittänyt sydämen sykevälivaihteluun perustuvan menetelmän ihmisen kehon toimintojen analysointiin. Yrityksen kehittämää menetelmää hyödynnetään työterveydenhuollon palveluissa, huippu-urheilussa valmennuksen tukena ja laitevalmistajien tuotteissa.

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa toimeksiantajalle laitekirjaston analysointiin analysointityökalu, jonka avulla voidaan visuaalisesti analysoida ja verrata erilaisten profiilitietojen ja mittaustietojen vaikutuksia kirjaston ulostuloihin. Opinnäytetyön tuotosta toimeksiantaja voi hyödyntää laitekirjaston ulostulojen analysoinnissa.

Opinnäytetyössä toteutettiin analytiikkatyökalu, joka koostui analytiikkatyökalu-sovelluksesta ja analytiikkatyökalu-kirjastosta. Analytiikkatyökalu-sovellus toteutettiin hyödyntäen Qt-sovelluskehystä ja kaavioiden esittämisessä Qt Charts -kirjastoa sekä suunnittelussa hyödynnettiin Qt model/view -arkkitehtuuria. Analytiikkatyökalu-kirjasto toteutettiin hyödyntäen C++-ohjelmointikielen STL-kirjastoa ja esikääntäjiä sekä suunnittelussa hyödynnettiin Bridge-suunnittelumallia.

Opinnäytetyön tuloksena oli analytiikkatyökalu, joka toteutti osan analytiikkatyökalun vaatimusmäärittelyn vaatimuksista. Analytiikkatyökalu-sovelluksella oli mahdollista esittää kaavioita analytiikkatyökalu-kirjaston ulostuloista. Analytiikkatyökalu-kirjastolla oli mahdollista muodostaa ajonaikainen yhteensopivuus laitekirjastolle. Analytiikkatyökalua toimeksiantaja voi hyödyntää erilaisten profiilitietojen, mittaustietojen ja laitekirjastojen analysoinnissa.

Avainsanat (asiasanat)
Qt, Qt-sovelluskehys, Qt Charts, Qt model/view -arkkitehtuuri, bridge-suunnittelumalli

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Kähkönen, Jani	Bachelor's thesis	December 2018
		Language of publication: Finnish
	Number of pages: 38	Permission for web publication: x

Title of publication
Firstbeat analytics tool

Degree programme
Software Engineering

Supervisor(s)
Rantala Ari, Huotari Jouni

Assigned by
Firstbeat Technologies Oy

Abstract

The thesis was assigned Firstbeat Technologies Oy, a welfare software company founded in 2002 in Jyväskylä. The company has developed a heart rate variation-based method for analyzing human body functions. The method developed by the company is used in occupational healthcare services, top-level sports coaching and device manufacturers' products.

The aim of thesis was to design and implement an application for the assignor to analyze the device library, which can be used for visually analyzing and comparing the effects of different profile data and measurement data on library outputs. The client can utilize the result of the thesis in analyzing the outputs of the device library since the application can detect possible deviations in the outputs of the library.

The implementation of the thesis was the analytics tool consisting of the analytics tool application and the analytics tool library. The analytics tool application was implemented utilizing the Qt application framework, the Qt Charts library was used to represent charts and the Qt model/view architecture was used in the design. The analytics tool library was implemented using the C++ programming language STL library and preprocessors. The Bridge design pattern was used in the design.

The result of the thesis was an analytics tool that implemented a part of the requirements of the analytics tool requirements definition. It was possible to present graphs of the outputs of the analytics tool library with the analytics tool application. The analytics tool library was able to form runtime compatibility with the device library. The analytics tool can be used by the assignor to analyze various profile information, measurement data, and device libraries.

Keywords/tags (subjects)
Qt, Qt framework, Qt Charts, Qt model/view architecture, bridge pattern

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

1 Johdanto

1.1 Tausta

Opinnäytetyön toimeksiantaja oli Firstbeat Technologies Oy, joka on vuonna 2002 Jyväskylässä perustettu hyvinvointialan ohjelmisto- ja palveluyritys. Yrityksen perustamista on edeltänyt laaja tutkimustyö urheilijoiden ylikunnon mittaamisessa Kilpa- ja huippu-urheilun tutkimuskeskuksessa (KIHU). Jyväskylän yliopistossa käynnistyi 2000-luvun alussa tutkimus, jonka tuloksena todettiin, että ihmisen kehon toimintaa ja stressiä voidaan mitata epäsuorasti sykevaihteluun perustuvien menetelmien avulla. Yritys on kehittänyt sykereaktioihin ja sykevaihtelun analysointiin perustuvaa teknologiaa, joka mahdollistaa tarkan mittaustuloksen ilman laboratorio-olosuhteita tai -laitteistoa. (Tarinamme n.d.)

Toimeksiantaja on kehittänyt sydämen sykevälivaihteluun perustuvan menetelmän ihmisen kehon toimintojen analysointiin. Menetelmän avulla voidaan parantaa suorituskykyä, kartoittaa ja kehittää hyvinvointia. Menetelmällä voidaan tuottaa henkilökohtaista ja täsmällistä tietoa liikunnan vaikutuksesta, stressin hallinnasta sekä levon palauttavista vaikutuksista. Menetelmän tuottamaa monipuolista tietoa elämäntavoista hyödynnetään laajalti työterveydenhuollon palveluissa, huippu-urheilussa valmennuksen tukena ja erilaisissa laitevalmistajien tuotteissa. (Yritys n.d.)

Toimeksiantajan kehittämän analytiikan perustana on sydämen syketietoa analysoiva laitekirjasto, joka muuttaa mitatun syketiedon monipuoliseksi ja ymmärrettäväksi palautteeksi ihmisen kehon toiminnasta. (Fysiologia n.d). Mitatun syketiedon pohjalta laitekirjasto tuottaa erilaisia ihmisen fysiologisia toimintoja tai ominaisuuksia kuvaavia analyysin tuloksia tai ulostuloja, kuten hapen- ja energiankulutus, harjoituksen jälkeinen lisääntynyt hapenkulutus, stressi- ja palautumisreaktiot sekä useat muut ominaisuudet. Toimeksiantajan analytiikkaan perustuvia ominaisuuksia käytetään useiden eri laitevalmistajien tuotteissa.

1.2 Tarkoitus ja tavoite

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa toimeksiantajalle sisäiseen käyttöön analytiikkatyökalu, jonka avulla voidaan analysoida ja verrata erilaisten laitekirjastojen ulostuloja hyödyntäen erilaisia profiilitietoja sekä mittaustietoja. Opinnäytetyön tavoitteena oli alustariippumaton analytiikkatyökalu, joka toimisi yleisimmillä käyttöjärjestelmillä ja esittäisi kaikki kirjaston ulostulot visuaalisesti kaavioina. Opinnäytetyön tulosta toimeksiantaja voisi hyödyntää laitekirjaston ulostulojen analysoinnissa, koska analytiikkatyökalulla voidaan analysoida kirjaston ulostuloja sekä verrata erilaisien kirjastojen ulostuloja. Analytiikkatyökalu mahdollistaa toimeksiantajalle poikkeamien havaitsemisen erilaisten laitekirjastojen ulostuloissa.

Opinnäytetyössä selvitettiin seuraavat kehittämistehtävät:

- Miten toteutetaan alustariippumattomuus työpöytäsovelluksessa?
- Miten toteutetaan yhteensopivuus sovelluksen ja erilaisten kirjastojen välillä?

1.3 Rajaus

Opinnäytetyö rajattiin analytiikkatyökalun suunnitteluun ja toteutukseen. Opinnäytetyön aihealue rajattiin käsittämään ainoastaan alustariippumattomia menetelmiä työpöytäsovellusten toteutuksessa C++-ohjelmointikielellä. Opinnäytetyön toteutukseen sisältyy kaavioiden esittämiseen tarkoitettuja menetelmiä, koska analytiikkatyökalun avulla oli tarkoitus esittää kaavioita laitekirjaston tuottamista ulostuloista. Opinnäytetyön toteutukseen sisältyy sovelluksen toiminnan kannalta vain välttämättömät käyttöliittymän elementit, koska käyttöliittymän suunnitelmat eivät olleet vielä valmiita. Opinnäytetyön aihe-alueeseen ei sisälly mittaustietojen lukemiseen tarkoitettuja menetelmiä erilaisista tiedostoformaateista.

2 Vaatimusmäärittely

Vaatimusmäärittelyn tarkoituksena on kartoittaa ja kuvata toteutettavan sovelluksen toiminnallisuuksia ja ominaisuuksia, jotka määritellään asiakkaan tai sidosryhmien asettamien vaatimusten pohjalta. Vaatimukset voidaan jaotella käyttäjä- ja järjestelmävaatimuksiin, jotka tulisi olla kuvattu selkeästi ja johdonmukaisesti. Käyttäjävaatimukset kuvaavat sovelluksen toiminnallisia ja ei-toiminnallisia vaatimuksia, joiden tulisi määrittää yksinkertaisesti sovelluksen ulkoista käyttäytymistä, joita kuvataan yleensä luonnollisella kielellä ja kaavioilla. Järjestelmävaatimukset kuvaavat sovelluksen käyttäytymistä ja toiminnallisia rajoitteita, joiden tulisi määrittää yksityiskohtaisesti sovelluksen sisäistä käyttäytymistä, jota kuvataan yleensä luonnollisella kielellä tai muodollisella formaatilla. (Sommerville 2010, 83.)

Analytiikkatyökalu-sovelluksen vaatimusmäärittelyssä arvioitiin ja määriteltiin sovellukselle kohdistuvia tavoitteita ja oletuksia sekä sen toiminnallisuutta, laatuominaisuuksia ja rajoituksia. Vaatimusmäärittelyssä kuvatut vaatimukset kuvattiin yksinkertaisten käyttäjätarinoiden avulla, jotka tehtiin loppukäyttäjän näkökulmasta. Käyttäjätarinat kuvaavat käyttäjän ja sovelluksen välisiä vuorovaikutuksia tietyn päämäärän saavuttamisessa, joita voidaan käyttää sovelluksen vaatimuksien kuvaamisessa. (Paakki, 2011.) Vaatimusmäärittelyn pohjana käytettiin edellisen sovelluksen toiminnallisuuksia, jotka nykyisen sovelluksen oli toteuttava. Vaatimusmäärittelyn sovellukselle teki opinnäytetyön toimeksiantaja, mutta sovelluksen vaatimukset tulivat loppukäyttäjiltä tai sidosryhmiltä (ks. liite 1).

Analytiikkatyökalu-sovelluksen vaatimusmäärittely määriteltiin ongelmakentän pohjalta, joka tarkoittaa teknistä tai fyysistä kenttää, jossa ongelma esiintyy. Ongelmakentän määrittämiseksi oli ymmärrettävä edellisessä sovelluksessa ilmenneet rajoitteet ja heikkoudet. (Paakki, 2011.) Rajoitteena sovelluksessa oli suorituskyky, koska sovelluksen toteutuksessa oli käytetty hidasta tekniikkaa. Heikkoutena sovelluksessa oli käytettävyys, koska laitekirjastojen ulostulojen vertaaminen erillisistä kaavioista oli vaikeaa. Heikkoutena sovelluksessa oli ylläpidettävyys, koska sovelluksella oli riippuvuuksia kirjastoihin, jotka eivät toimineet ongelmitta kaikilla alustoilla. Opinnäytetyössä tehtävänä oli suunnitella ja toteuttaa tässä ongelmakentässä toimiva sovellus, joka ratkaisee tämän ongelman.

Analytiikkatyökalu-sovelluksen vaatimusmäärittelyssä määriteltiin sovellukselle toiminnalliset vaatimukset, jotka kuvaavat, mitä palveluita sovelluksen on tarjottava ja kuinka sovellus reagoi tiettyihin syötteisiin sekä kuinka sovellus toimii tietyissä tilanteissa. (Sommerville 2010, 85). Sovelluksen tuli pystyä tuottamaan kaaviot kaikista laitekirjaston ulostuloista, joista olisi helppo ja nopea huomata mahdolliset eroavaisuudet ulostuloissa. Sovelluksen tuli pystyä lukemaan mittaustietoja erilaisista tiedostoformaateista, joista olisi mahdollista syöttää mittaustietoja analyyseille. Sovelluksella tuli pystyä lukemaan profiilitietoja CSV-tiedostoformaatista, josta sovelluksen olisi mahdollista syöttää profiilitietoja analyyseille.

Analytiikkatyökalu-sovelluksen vaatimusmäärittelyssä määriteltiin sovellukselle eitoiminnalliset vaatimukset, jotka kuvaavat, mitkä ovat sovelluksen tarjoamien palveluiden tai toimintojen rajoitteet, kuten sovelluksen siirrettävyys, yhteensopivuus ja ylläpidettävyys. (Sommerville 2010, 87). Sovelluksen tuli olla siirrettävissä erilaisille käyttöjärjestelmille, koska sovellusta oli tarkoitus käyttää ainakin Windows-, Linux- ja OSX-käyttöjärjestelmissä. Sovelluksen tuli olla yhteensopiva erilaisille laitekirjastoille, koska sovellusta oli tarkoitus käyttää erilasten laitekirjastojen analysoinnissa. Sovelluksen tuli olla helposti ylläpidettävä, koska sovellusta oli tarkoitus päivittää tukemaan tulevia laitekirjastojen toiminnallisuuksia.

Analytiikkatyökalu-sovellukselle määritetyt vaatimukset:

1. Tulosten tallentaminen CSV-tiedostoon
2. Tulosten esittäminen kaavioissa
3. Kaavion zoomaus
4. Toimintaympäristö
5. FIT-tiedostoformaatin yhteensopivuus
6. SDF-tiedostoformaatin yhteensopivuus
7. FBE-tiedostoformaatin yhteensopivuus
8. Laitekirjaston valinta vaihtoehdot
9. Profiilitietojen lukeminen tiedostosta
10. Profiilitietojen asettaminen sallituilla tiedoilla
11. Profiilitietojen asettaminen sallituilla alueilla
12. Analyysin parametrien valitseminen
13. Analyysin parametrien päivittäminen
14. Analyysin sisäisten muuttujien tallentaminen tuloksiin

3 Menetelmät

Opinnäytetyön menetelmävalintojen lähtökohdat määritteli analytiikkatyökalun vaatimusmäärittely, jonka pohjalta valittiin toteutuksessa käytettävät menetelmät. Opinnäytetyön toteutuksessa käytettäväksi menetelmäksi valittiin Qt-sovelluskehys, koska sovelluskehys mahdollisti sovelluksessa kaavioiden esittämisen sekä sovelluksen siirrettävyyden yleisimmille käyttöjärjestelmille. Sovelluksen kaavioiden esittämisen mahdollisti Qt Charts -moduuli, jonka avulla voitiin esittää tarvittavat kaaviot. Sovelluksen siirrettävyyden yleisimmille käyttöjärjestelmille mahdollisti Qt-sovelluskehys ja Qmake-työkalu, joiden avulla sovellus oli käytettävissä ja käännettävissä Linux-, OSX- ja Windows-käyttöjärjestelmissä. Sovelluksen ylläpidettävyys varmistettiin valitsemalla Qt-sovelluskehys, koska sovelluskehys oli ennestään tuttu menetelmä toimeksiantajalle työskenteleville sovelluskehittäjille sekä sovelluskehys mahdollisti kaikki vaaditut ominaisuudet analytiikkatyökalulle.

3.1 Qt-sovelluskehys

Qt-sovelluskehys on alustariippumattomien GUI-sovellusten kehittämiseen tarkoitettu kirjasto, joka antaa sovelluskehittäjille mahdollisuuden käyttää vain yhtä alustariippumatonta koodipohjaa sovelluksen toteutuksessa. (Blanchette & Summerfield 2008, xi). Qt-sovelluskehyksen avulla voidaan kehittää sovelluksia työpöytä-, sulautettu- ja mobiiliympäristöihin, jotka toimivat Linux-, OSX-, Windows-, VxWorks-, QNX-, Android-, iOS-, BlackBerry- ja Sailfish OS ja muilla ohjelmisto- ja laitealustoilla. (About Qt n.d). Qt-sovelluskehyksen avulla GUI-sovellukset toteutetaan Widget-moduulilla, jonka avulla graafiset käyttöliittymät voidaan kirjoittaa C++-ohjelmointikielellä tai toteuttaa Qt Designer -työkalulla. Qt sisältää interaktiivisen graafisen työkalun nimeltä Qt Designer, joka toimii koodigeneraattorina widget-moduuliin perustuvissa GUI-sovelluksissa.

Qt-sovelluskehyksellä voidaan toteuttaa alustariippumattomia graafisia käyttöliittymiä sovelluksille, mutta sovellusten käyttöliittymät eivät välttämättä näytä samanlaisilta eri käyttöjärjestelmissä, koska käyttöjärjestelmissä on erilainen visuaalisen muotoilu. Qt-sovelluskehys sisältää useita moduuleita, joita voidaan käyttää sovellusten kehittämisessä (ks. taulukko 1).

Taulukko 1. Qt essentials -moduulit

Moduuli	Kuvaus
Qt Core	Perustoiminnot moduuleille
Qt GUI	Graafiset käyttöliittymät
Qt Multimedia	Ääni-, video-, radio- ja kameratoiminnot
Qt Multimedia Widgets	Multimedia-toiminnot widgeteillä
Qt Network	Tietoverkko-toiminnot
Qt QML	Sovelluskehys QML-ohjelmointikielelle
Qt Quick	Graafiset käyttöliittymät Qt Quick-sovelluksissa
Qt Quick Controls 2	Graafisen käyttöliittymän elementit Qt Quick-sovelluksissa
Qt Quick Dialogs	Järjestelmän valintaikkunoiden luominen ja vuorovaikutus Qt Quick-sovelluksissa
Qt Quick Layouts	Käyttöliittymän elementtien asettelu Qt Quick-sovelluksissa
Qt Quick Test	Yksikkötestaaminen Qt Quick-sovelluksissa
Qt SQL	SQL-tietokantojen integrointi sovelluksissa
Qt Test	Yksikkötestaaminen Qt-sovelluksissa
Qt Widgets	Graafisen käyttöliittymän laajentaminen widgeteillä Qt-sovelluksissa

(Qt Essentials n.d.)

Qt-sovelluskehys on saatavana kaupallisella lisenssillä ja vapaiden ohjelmistojen lisenssillä. Kaupallisen Qt Group -yrityksen lisenssin mukaisesti lisensoitu Qt-sovelluskehys on sopiva sovellusten kehittämisessä, jos lähdekoodia ei haluta jakaa kolmansille osapuolille tai muuten noudattaa GNU LGPL version 3 -lisenssin ehtoja. Vapaiden ohjelmistojen lisenssin mukaisesti lisensoitu Qt-sovelluskehys on sopiva sovellusten

kehittämisessä, jos on mahdollista noudattaa GNU LGPL version 3 -lisenssin tai GNU GPL version 3 -lisenssin ehtoja. (Qt Licensing n.d.) Qt-sovelluskehys on saatavana kaupallisella sekä vapaan ohjelmiston lisensseillä, joista kaupallinen lisenssi on maksullinen ja vapaan ohjelmiston lisenssit rajoittavat julkaisua, joka täytyy tehdä enemmän tai vähemmän rajoittavilla ehdoilla.

Qt core-moduuli on ainoa Qt-sovelluskehyksen pakollinen moduuli, joka sisältää muiden moduulien tarvitsemia toiminnallisuuksia, kuten Meta-object system -ominaisuus, signal and slot -mekanismi, tietorakenteet ja I/O-ominaisuudet. (Qt Core 2018). Meta-object system -ominaisuus mahdollistaa viestintämenetelmän, ajonaikaisen tyyppitiedon ja dynaamiset ominaisuudet. (The Meta-Object System 2018). Signal and slot -mekanismi mahdollistaa olioiden välisen viestinnän, jota voidaan käyttää graafisissa käyttöliittymissä sijaitsevien widget-elementtien viestinnässä. (Signals & Slots 2018). Tietorakenteet mahdollistavat erilaisia assosiatiivisia template-pohjaisia säiliöitä, kuten QList, QLinkedList, QVector, QStack ja QQueue sekä QMap, QMultiMap, QHash, QMultiHash ja QSet. (Container Classes 2018). I/O-ominaisuudet mahdollistavat syötteet ja tulosteet ulkoisista resursseista, kuten laitteista, prosesseista ja tiedostoista. (Input/Output and Networking 2018).

Qt Core -moduuli on saatavana kaupallisella Qt Group-yrityksen lisenssillä ja vapaiden ohjelmistojen lisenssillä. Qt 5.4 version jälkeen vapaiden ohjelmistojen lisenssit ovat GNU Lesser General Public License version 3 -lisenssi ja GNU General Public License version 2 -lisenssi. (Qt Core 2018.)

Meta-Object Compiler on Qt-sovelluskehyksen taustalla toimiva esikääntäjä, jota käytetään laajentamaan C++-ohjelmointikielen ominaisuuksia Qt-sovelluskehyksen sisäänrakennetuilla ominaisuuksilla. Ennen käännösvaihetta esikääntäjä Meta-Object Compiler parsii lähdekooditiedostot, jotka on kirjoitettu Qt-sovelluskehyksen laajennetulla C++-ohjelmointikielellä, minkä jälkeen Meta-Object Compiler luo lähdekooditiedostoista standardinmukaiset C++-lähdekooditiedostot. Meta-Object Compilerin luomat C++-lähdekooditiedostot käännetään ja linkitetään. Sovellukset ja kirjastot, jotka käyttävät Meta-Object Compileria, voidaan kääntää millä tahansa standardinmukaisella C++-kääntäjällä, kuten Clang, GCC, ICC, MinGW ja MSVC. (About Qt n.d.)

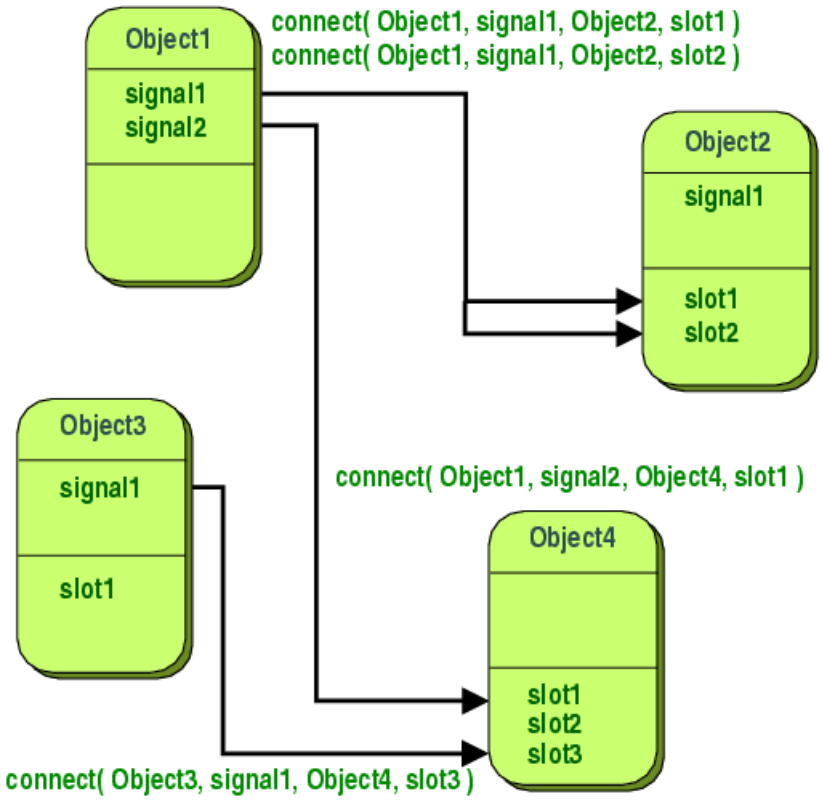
Qt-sovelluskehys ei ole puhdas C++-kirjasto, koska se edellyttää erillisiä käännösvaiheita, mikä tekee sovelluksen kääntämisestä monimutkaisen verrattuna muihin kirjastoihin.

Meta-Object System -ominaisuus on tarkoitettu Qt-sovelluskehys kohtaisten laajennusten tukemiseen sovelluksissa, kuten ajoaikainen tyypitys, dynaaminen ominaisuusjärjestelmä ja olioiden välinen viestintämenetelmä. Meta-Object System -ominaisuus perustuu kolmeen asiaan: QObject-luokka määrittää kantaluokan olioille, jotka voivat hyödyntää Meta Object System -ominaisuuksia. Q_OBJECT-makron määrittäminen luokkamäärittelyssä ottaa käyttöön metaominaisuudet, minkä jälkeen Meta-Object Compiler toimittaa QObject-alaluokille tarvittavan koodin toteuttamaan metaominaisuudet. Meta-Object Compiler etsii C++-lähdekooditiedoston luokkamäärittelyistä Q_OBJECT-makroja, joista tuotetaan toiseen C++-lähdekooditiedostoon luokalle metaobjektikoodi. (The Meta-Object System n.d.) Qt-sovelluskehyksen ominaisuudet ovat käteviä, mutta muut kehitysympäristöt voivat pitää Qt-syntaksia virheinä, koska nämä eivät ymmärrä Qt-sovelluskehyskohtaista syntaksia, joka voi pakottaa sovelluskehittäjän käyttämään Qt creator -kehitysympäristöä.

Signal and slot -mekanismi on tarkoitettu graafisissa käyttöliittymissä sijaitsevien widget-elementtien väliseen tapahtumankäsittelyyn, jonka avulla widget-elementit voivat lähettää signaaleja, jotka sisältävät tapahtumatietoja, joita muut widget-elementit vastaanottavat sloteilla. Esimerkiksi kun käyttäjä painaa sulje-painiketta, halutaan kutsua kyseisen ikkunan sulje-toimintoa. Muissa sovelluskehyksissä tämä toiminnallisuus on toteutettu yleensä callback-funktioiden avulla. Callback-funktio on osoitin toimintoon, kun prosessointitoiminnon halutaan ilmoittavan tapahtumasta, siirretään osoitin käsittelyfunktiosta toiseen käsittelyfunktioon, minkä jälkeen käsittelyfunktio kutsuu sopivana ajankohtana callback-funktiota. Vaikka tätä menetelmää käytetään muissa samanlaisissa sovelluskehyksissä, callback-funktiot voivat olla ongelmallisia, koska callback-funktion argumenttien tyyppioikeellisuutta on vaikea varmistaa. (Signals & Slots n.d.)

Signal and slots -mekanismi koostuu signaaleista ja sloteista (ks. kuvio 1). Signaali lähetetään aina silloin, kun jokin tietty tapahtuma ilmenee. Widget-elementeillä on useita ennalta määriteltyjä signaaleja. Widget-elementeistä voidaan tehdä alaluokkia, joihin voidaan lisätä uusia signaaleja tapahtumista, joista ollaan kiinnostuneita.

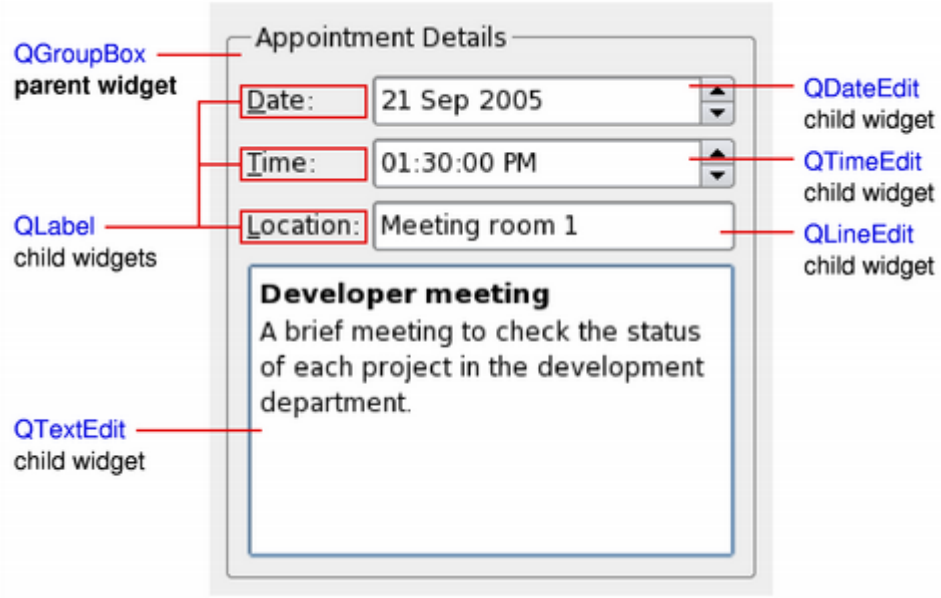
Slot on tapahtuman vastaanottava toiminto, jota kutsutaan vastauksena tiettyyn tapahtumaan. Widget-elementeillä on useita ennalta määriteltyjä slotteja. Widget-elementeistä voidaan tehdä alaluokkia ja lisätä uusia slotteja, joissa voidaan käsitellä tapahtumia, joista ollaan kiinnostuneita. (Signals & Slots n.d.)



Kuvio 1. Qt signal and slot -mekanismi

Signal and slots -mekanismi on tyyppiturvallinen, koska funktioiden allekirjoitukset ovat yhteensopivia, koska kääntäjä voi auttaa havaitsemaan tyypin epäselvyydet käytettäessä funktio-osoitinsyntaksia. Merkkijonopohjainen SIGNAL- ja SLOT-syntaksi tunnistaa tyypin yhteensopimattomuudet ajonaikaisesti. Signaalit ja slotit ovat löyhästi kytkettyjä, koska signaalin lähettävä olio ei tunne tai välitä siitä, mikä olio vastaanottaa signaalin. Qt-sovelluskehyksen signal and slot -mekanismi varmistaa, että signaali yhdistetään aina oikeaan slottiin ja slot-funktiota kutsutaan aina oikeilla parametreilla. Signaalit ja slotit voivat ottaa vastaan minkä tahansa määrän ja minkä tahansa tyyppisiä argumentteja. (Signals & Slots n.d.)

Qt-sovelluskehyksen Widget -moduuli tarjoaa useita käyttöliittymissä käytettäviä elementtejä, joita voidaan käyttää työpöytäsovelluksien käyttöliittymien toteutuksessa (ks. kuvio 2). Widget-elementit ovat Qt-sovelluskehyksen ensisijaisia elementtejä työpöytäsovellusten käyttöliittymien toteutuksessa. Widget-elementit voivat esittää käyttäjälle tietoa ja tilatietoja, vastaanottaa käyttäjän syötteitä sekä tarjota säiliön toisille widget-elementeille. Widget-elementtiä, jolla ei ole isäntä widget-elementtiä, kutsutaan ikkunaksi. Widget-elementtejä voidaan luoda perimällä QWidget-luokka tai sopiva QWidget-alaluokka, jonka virtuaaliset tapahtumankäsittelijät voidaan ylikirjoittaa. Widget-elementit integroituvat hyvin taustalla olevaan käyttöjärjestelmän käyttöliittymän teemaan, jotka tarjoavat natiivin ulkoasun Linux-, OSX ja Windows-käyttöjärjestelmissä. (Qt Widgets n.d.)



Kuvio 2. Qt widget -elementit

Qt Widgets -moduuli on saatavana kaupallisella Qt Group-yrityksen lisenssillä ja vapaiden ohjelmistojen lisenssillä. Qt 5.4 version jälkeen vapaiden ohjelmistojen lisenssit ovat GNU Lesser General Public License version 3 -lisenssi ja GNU General Public License version 2 -lisenssi. (Qt Widgets 2018.)

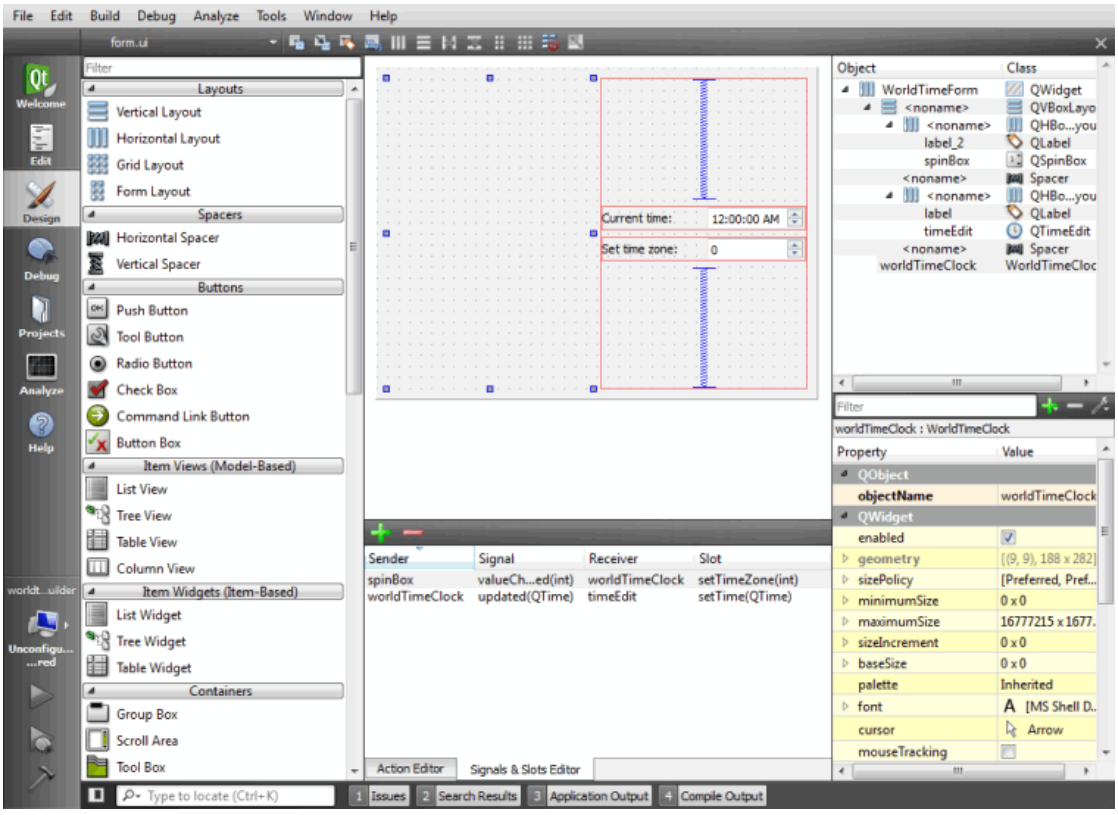
Qt-sovelluskehyksen Charts -moduuli tarjoaa useita käyttöliittymässä käytettäviä kaaviokomponentteja, jotka hyödyntävät Qt Graphics View Framework -kirjastoa, jonka avulla kaavioita voidaan integroida erilaisiin käyttöliittymiin (ks. kuvio 3). Qt Charts -moduulin kaavioita voidaan käyttää sovelluksissa QWidgets- tai QGraphicsWidget-tyyppeinä, joiden avulla sovelluskehittäjät voivat toteuttaa erityyppisiä kaavioita. QChart-luokka ohjaa kaavioissa käytettävien sarjojen ja muiden elementtien graafista esitystä, kuten otsikoita ja akseleita. QChart-luokan kaavioita voidaan esittää joko QChart-luokan tai QChartView-luokan avulla. Kaavioiden ulkoasua voidaan muokata erilaisilla teemoilla ja väreillä sekä piilottamalla kaavion osia tai tekemällä kaavioihin animaatioita. (Qt Charts n.d.)

Kuvio 3. Qt Charts -kaaviot

Qt Charts -moduuli on saatavana kaupallisella Qt Group-yrityksen lisenssillä ja vapaiden ohjelmistojen GNU General Public License version 3 -lisenssillä. (Qt Charts 2018.)

3.2 Qt creator -kehitysympäristö

Qt Creator on integroitu kehitysympäristö, joka tarjoaa sovelluskehittäjille työkaluja sovellusten suunnitteluun ja toteuttamiseen (ks. kuvio 4). Qt Creator -kehitysympäristö on suunniteltu sovellusten ja käyttöliittymien kehittämiseen useille eri työpöydille, sulautetuille ja mobiililaitealustoille sekä käyttöjärjestelmille. Qt Creator -kehitysympäristö tarjoaa työkaluja erilaisten tehtävien toteuttamiseen koko sovelluksen kehitys elinkaaren ajaksi luomalla projektin ja tekemällä sovellukselle käyttöönoton kohdejärjestelmälle ja laitteistoalustalle. Qt Creator -kehitysympäristö on saatavana Linux-, OSX- ja Windows-käyttöjärjestelmille. Qt Creator-kehitysympäristö tarjoaa koodin täydentäjän, syntaksin korostuksen, integroidun ohjetoiminnon, debuggerin sekä integroinnin yleisimmille versionhallintajärjestelmille. (IDE n.d; IDE Overview n.d.)



Kuvio 4. Qt Creator -kehitysympäristö

Qt Creator -kehitysympäristö on integroitu työkaluihin, joita käytetään alustariippumattomien sovelluksen rakentamisessa, kuten qmake, qbs, cmake ja autotools. Qt Creator -kehitysympäristöön voidaan tuoda projekteja, joiden rakentamisessa käytettäviä vaiheita ja komentoja voidaan hallita kehitysympäristössä. Qt Creator -kehitysympäristö tukee Qt-sovellusten ajamista työpöytäympäristöissä tai laitteissa. Kehitysympäristö tarjoaa tuen Qt-sovellusten ajamiseen ja käyttämiseen eri kohdeympäristöissä, joka voidaan tehdä käyttämällä erilaisia kääntäjiä, debuggereita. Qt Creator-kehitysympäristössä voidaan määrittää projektin asetukset, kuten työkalut, laitteen tyyppi ja muut asetukset, joita käytetään sovelluksen rakentamisessa ja ajamisessa. (Building and Running n.d; IDE Overview n.d.)

Qt Creator -kehitysympäristö sisältää tekstieditorin, jonka avulla lähdekoodin kirjoittaminen, muokkaaminen ja navigointi ovat yksinkertaista. Qt Creator -kehitysympäristön tekstieditori ymmärtää C++-ohjelmointikieltä koodina, mikä mahdollistaa ohjelmistokehittäjille hyödylliset ominaisuudet sovellusten kehittämisessä, kuten koodin semanttisen korostuksen, koodin syntaksin tarkistuksen, koodin täydentämisen ja koodin refaktorointi toiminnot. Qt Creator -kehitysympäristön avulla voidaan myös rakentaa ja ajaa sovelluksia. Qt Creator -kehitysympäristö sisältää hakutoiminnon, jonka avulla voidaan etsiä tiedostoja avoimesta projektista tai tiedostojärjestelmästä. (Coding 2018; IDE Overview 2018.)

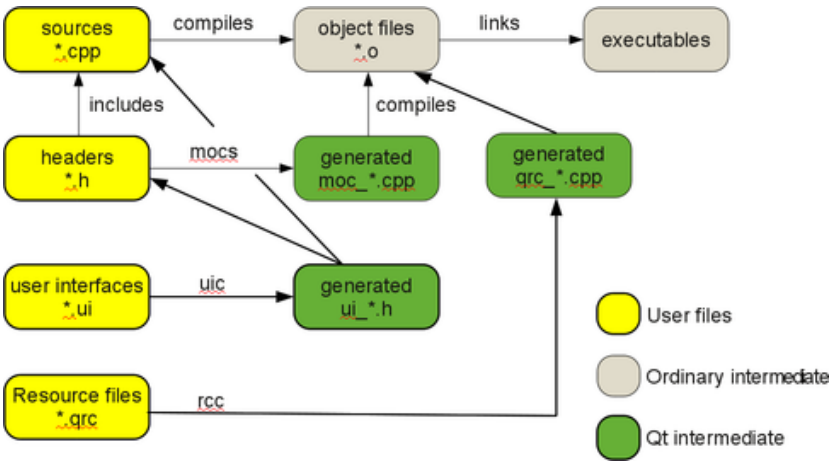
Qt Creator -kehitysympäristö sisältää kehitysympäristöön integroidut visuaaliset editorit Qt Quick- ja widget-pohjaisten sovellusten käyttöliittymien luomiseen. Qt Designer -työkalulla tehdyt Widgetit integroidaan ohjelmoituun koodiin käyttäen Qt-sovelluskehyksen signal and slots- mekanismia, jonka avulla voidaan määrittää käyttäytyminen kaikille käyttöliittymässä oleville graafisille elementeille. Kaikkia Qt Designer -työkalulla asetettuja elementtejä voidaan muuttaa myös dynaamisesti koodin sisällä. Qt Designer -työkalulla voidaan lisätä myös omia widget-elementtejä, jotka käännetään ensin dynaamiseksi kirjastoksi ja ladataan ajonaikaisesti. (Designing User Interfaces 2018; IDE Overview 2018.)

Qt creator -kehitysympäristö on integroitu useisiin ulkopuolisiin debuggereihin, kuten GNU Symbolic Debugger (GDB), Microsoft Console Debugger (CDB) ja sisäiseen JavaScript-debuggeriin. Qt Creator -kehitysympäristössä sovelluksen tilaa voidaan tarkkailla Qt Creator -kehitysympäristö Debug-tilassa, jonka avulla voidaan käyttää

toimintoja, kuten eteneminen sovelluksen läpi eteneminen rivi kerrallaan, sovelluksen pysäyttäminen haluttuun kohtaan, muuttujien tarkastelu ja muokkaaminen. Qt Creator -kehitysympäristö on integroitu Valgrind-analysointityökaluun, jonka avulla voidaan havaita sovelluksen muistivuodot ja profiloida toimintojen suoritusta. Qt Creator -kehitysympäristö on integroitu Qt Test ja Google C++-testauskehyksiin sovellusten ja kirjastojen yksikkötestausta varten. (Testing 2018; IDE Overview 2018.)

3.3 Qmake-työkalu

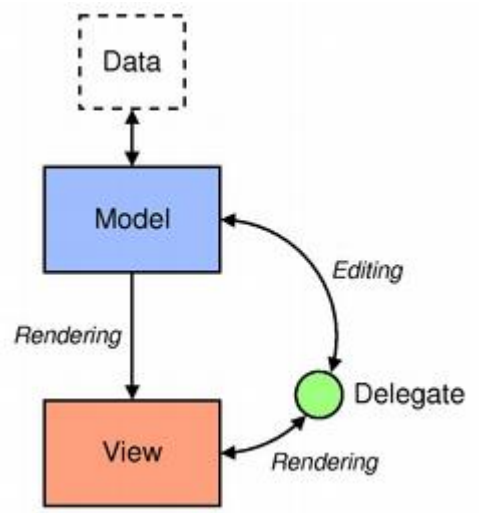
Qmake-työkalu on tarkoitettu yksinkertaistamaan sovelluksen kääntämisprosessia eri alustoilla. Qmake-työkalu automatisoi makefile-tiedostojen luomisen siten, että makefile-tiedostot voidaan luoda vain muutamalla rivillä Qmake-komentoja. Qmake-työkalua voidaan hyödyntää missä tahansa ympäristössä. Qmake-työkalu tuottaa makefile-tiedoston Qmake-projektitiedoston tietojen perusteella, jotka on luonut sovelluskehittäjä. Qmake-projektitiedostot ovat yleensä yksinkertaisia, mutta monimutkaisempia sovellus projekteja varten voidaan luoda monimutkaisempia projektitiedostoja. Qt-sovelluskehyksen tueksi Qmake-työkalu sisältää lisäominaisuuksia, jotka sisällyttävät käännösprosessiin automaattisesti myös Meta-Object Compiler- ja User Interface Compiler-sääntöjä (ks. kuvio 5). Qmake-työkalu voi luoda projektitiedostot myös Microsoft Visual Studio -ohjelmointiympäristöä varten ilman, että sovelluskehittäjää vaaditaan muuttamaan projektitiedostoja. (qmake Manual 2018.)



Kuvio 5. Qt-sovelluksen käännösprosessi

3.4 Model/view -arkkitehtuuri

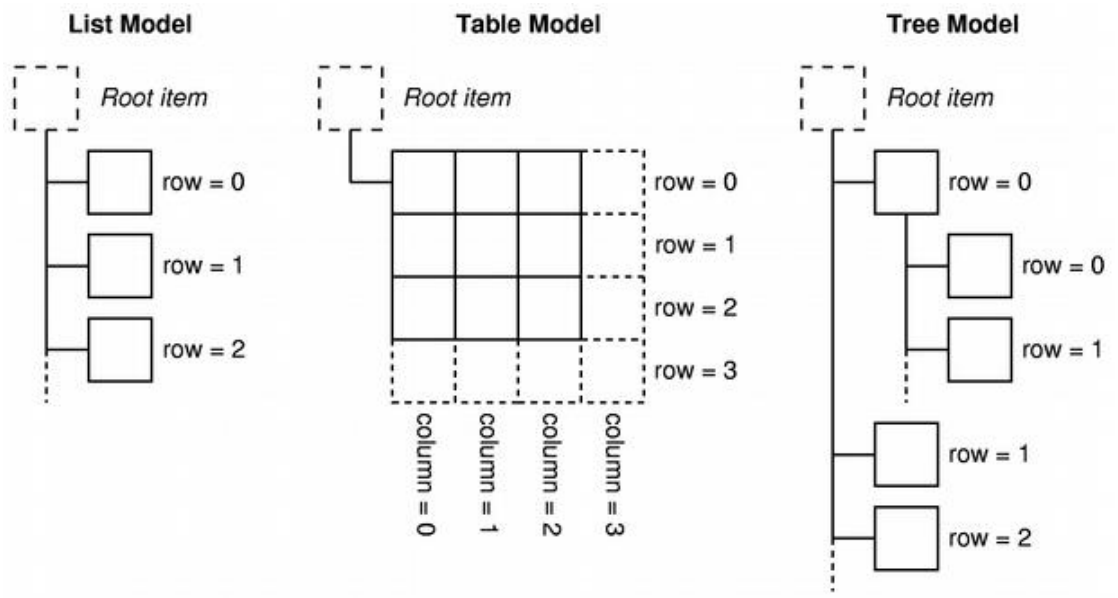
Qt-sovelluskehys sisältää useita näkymä luokkia, jotka käyttävät model/view -arkkitehtuuria hallitakseen tiedon ja esityksen välistä suhdetta. Tämä arkkitehtuurin toiminnallisuuden erottelu antaa suuremman joustavuuden mukauttaa elementtien esitystä ja tarjota standardi rajapinta, joka mahdollistaa erilaisten tietolähteiden käytön näkymissä. Model/view -arkkitehtuuri voidaan jakaa kolmeen ryhmään, jotka ovat model, view ja delegate (ks. kuvio 6). Jokainen näistä komponenteista on määritelty abstrakteilla luokilla, jotka tarjoavat yleisen rajapinnan ja joissakin tapauksissa oletus toiminnallisuudet. Abstrakteista luokista on tarkoitus tehdä alaluokkia, joiden avulla saadaan toteutettua kehitettävien komponenttien odottamat toiminnallisuudet. Abstraktit luokat mahdollistavat myös mukautettujen komponenttien toteuttamisen. (Model/View Programming 2018.)



Kuvio 6. Qt model/view -arkkitehtuuri

Model/view -arkkitehtuurissa model-komponentti tarjoaa standardin rajapinnan, jonka avulla view- ja delegate-komponentit pääsevät käsiksi tietoihin. Qt-sovelluskehyksessä standardi rajapinta on määritelty QAbstractItemModel-luokalla riippumatta siitä, miten tieto on tallennettu taustalla olevassa tietorakenteessa. QAbstractItemModel-alaluokat edustavat tietoa hierarkkisina rakenteina (ks. kuvio 7). View-kom-

ponentit käyttävät tätä käytäntöä päästäkseen käsiksi model-komponentin sisältämiin tietoihin, mutta view-komponentteja ei ole rajoitettu, kuinka tämä tieto esitetään käyttäjälle. Model-komponentit myös ilmoittavat jokaiselle liitetylle näkymälle tietojen muutoksista signal and slot -mekanismin avulla. (Model Classes n.d.)



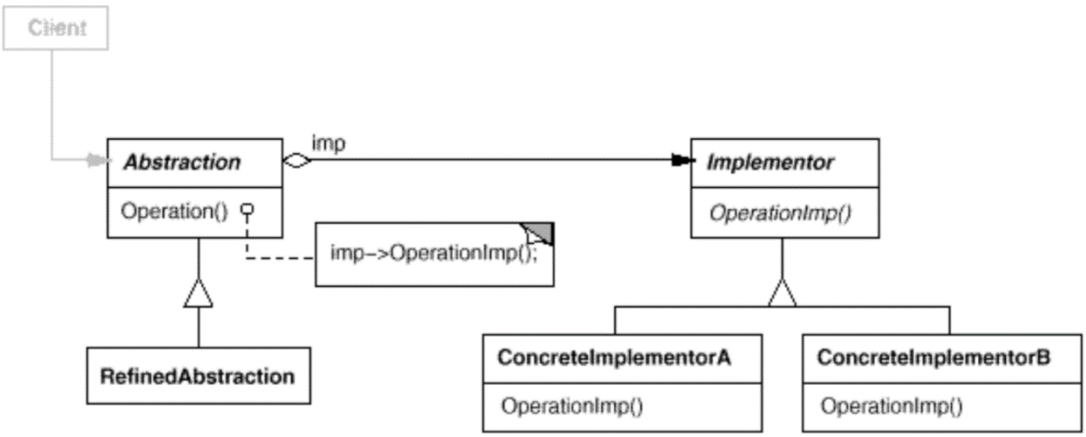
Kuvio 7. Qt model/view -arkkitehtuurin model-komponentit

Model/view -arkkitehtuurissa view-komponentti saa tiedot model-komponentilta, minkä jälkeen view-komponentti esittää tiedon käyttäjälle. Tietojen esittämistavan ei tarvitse muistuttaa model-komponentin toimittaman tiedon muotoa, koska se voi olla täysin eri muodossa, johon se on tallennettu taustalla olevassa tietorakenteessa. Tiedon ja esityksen erottaminen mahdollistetaan käyttämällä standardia model-komponentin QAbstractItemModel-luokkaa, standardia view-komponentin QAbstractItemView-luokkaa ja model-komponentin indeksejä, jotka edustavat tietoa yleisellä tavalla. Model-komponenteista saatujen tietojen yleistä asettelua hallitsevat tyypillisesti View-komponentit, jotka voivat esittää yksittäisiä tietoa tai käyttää delegate-komponentteja käsittelemään sekä esitys- ja muokkaustoiminnot. (View Classes n.d.)

Model/view -arkkitehtuurissa ei ole täysin erillistä komponenttia vuorovaikutuksen hallintaan käyttäjän kanssa. Delegate-komponenttit hoitavat vuorovaikutuksen, joka tarjoaa syötteiden käsittelytoiminnallisuudet sekä ne vastaavat yksittäisen tiedon esittämisestä tietyissä näkymissä. Standardi rajapinta delegate-komponenttien hallintaan on määritelty QAbstractItemDelegate-luokassa. Delegate-komponenttien odotetaan pystyvän esittämään oman sisältönsä toteuttamalla paint- ja sizeHint-toiminnallisuudet. Yksinkertaiset widget-pohjaiset delegate-komponentit voivat olla aliluokkia QItemDelegate-luokalle QAbstractItemDelegate-luokan sijasta ja toteuttaa QItemDelegate-luokan oletustoiminnallisuudet. Muokkaustoiminnallisuus delegate-komponenteille voidaan toteuttaa käyttämällä widget-elementtejä hallitsemaan muokkausprosessia tai käsittelemällä tapahtumat suoraan. (Delegate Classes n.d.)

3.5 Bridge-suunnittelumalli

Bridge-suunnittelumalli on sovelluskehityksessä käytetty rakenteellinen suunnittelumalli, jonka tarkoituksena on erottaa abstraktio sekä toteutus siten, että nämä kaksi voivat vaihdella itsenäisesti (ks. kuvio 8). Bridge-suunnittelumalli on hyödyllinen, kun luokan rajapinta pysyy samana ja luokan toiminnallisuudet vaihtelevat usein. Luokkaa voidaan ajatella abstraktioksi ja luokan toimintoja toteutukseksi. Bridge-suunnittelumalli hyödyntää olio-ohjelmoinnin ominaisuuksia, kuten kapselointia ja koostamista. Lisäksi voidaan käyttää perintää erottamaan vastuita erillisiin luokkiin. Suunnittelumallin nimi on Bridge, koska se muodostaa sillan abstraktion ja toteutuksen välille. (Gamma, Helm, Johnson & Vlissides, 1994.)



Kuvio 8. Bridge-suunnittelumalli

Bridge-suunnittelumallia suositellaan käytettäväksi seuraavissa tapauksissa: Halutaan välttää pysyvä sitoutuminen abstraktion ja toteutuksen välillä. Näin voi olla esimerkiksi silloin, kun toteutus on valittava tai vaihdettava suoritusaikaan. Abstraktion ja toteutuksen tulee olla laajennettavissa alaluokkiin, koska suunnittelumallin avulla voidaan yhdistää eri abstraktiot ja toteutukset sekä laajentaa niitä itsenäisesti. Muutokset abstraktion toteutuksessa eivät saa vaikuttaa asiakasohjelmaan, eli koodia ei pitäisi joutua kääntämään uudestaan. Halutaan piilottaa abstraktion toteutus kokonaan asiakasohjelmalta. (Gamma ym, 1994.)

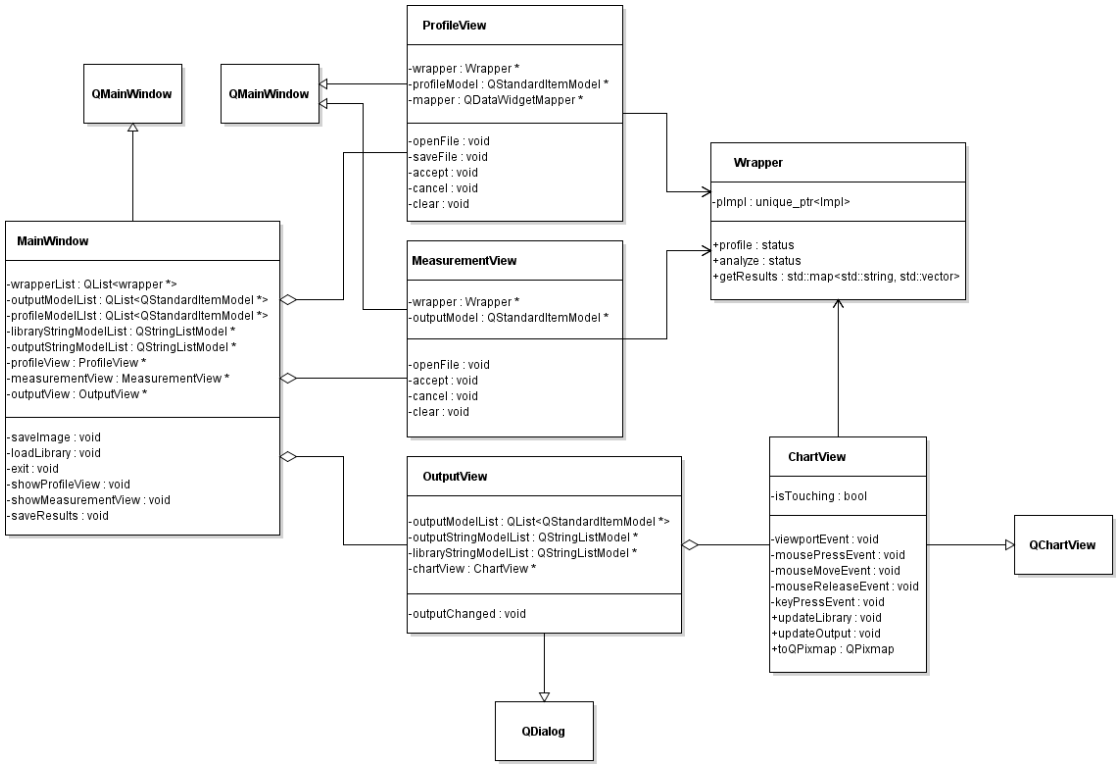
4 Toteutus

Analytiikkatyökalu-sovellus toteutettiin hyödyntämällä Qt-sovelluskehystä ja kaavioiden esittämisessä käytettiin Qt Charts -moduulia sekä suunnittelussa hyödynnettiin Qt model/view -arkkitehtuuria. Qt-sovelluskehys mahdollisti sovelluksen toteuttamisen alustariippumattomasti. Analytiikkatyökalu-kirjasto toteutettiin hyödyntämällä C++-ohjelmointikielen STL-kirjastoa ja esikääntäjiä sekä suunnittelussa hyödynnettiin Bridge-suunnittelumallia. Bridge-suunnittelumalli mahdollisti yhteensopivuuden erilaisille laitekirjastoille. Analytiikka-työkalun toteutuksessa hyödynnettiin Qt Creator -kehitysympäristöä, Qt Designer -suunnittelutyökalua, Qmake-työkalua ja C++-ohjelmointikieltä.

Analytiikkatyökalun käännösprosessissa hyödynnettiin Qmake-työkalua, joka on tarkoitettu yksinkertaistamaan sovelluksen käännösprosessia eri alustoilla. Qmake-työkalun avulla voidaan automatisoida makefile-tiedostojen luominen Qmake-projektitiedostoilla. Projektitiedostossa on määritelty sovelluksen käännösprosessissa käytettävä projektin tyyppi, moduulit ja ohjelmointikieli, sisällytettävät tiedostosijainnit, kirjastot, otsikko- ja lähdekooditiedostot sekä käyttöliittymä. Analytiikkatyökalu-sovelluksen projektitiedostossa on määritelty projektin tyypiksi sovellus, käytettäviksi moduuleiksi Charts- ja Widget-moduuli, ohjelmointikieleksi C++11, sisällytettäväksi tiedostosijainniksi analytiikkatyökalu-kirjaston otsikkotiedostot ja Qt Designer-työkalun UI-tiedosto. Analytiikkatyökalu-kirjaston projektitiedostossa on määritelty projektin tyypiksi kirjasto, sisällytettäväksi tiedostosijainniksi laitekirjaston otsikkotiedostot ja kirjastoksi staattinen laitekirjasto.

4.1 Analytiikkatyökalu-sovellus

Analytiikkatyökalu-sovelluksen tehtävänä oli esittää kaavioita analytiikkatyökalu-kirjaston ulostuloista. Analytiikkatyökalu-sovelluksen toteutuksessa käytettiin Qt-sovelluskehystä ja kaavioiden esittämisessä Qt Charts -kirjastoa sekä suunnittelussa hyödynnettiin Qt model/view -arkkitehtuuria. Analytiikkatyökalu-sovelluksen luokkakaavio on esitetty kuviossa 9. Qt-sovelluskehys mahdollisti GUI-työpöytäsovelluksen toteuttamisen alustariippumattomasti, jonka avulla analytiikkatyökalu-sovelluksen toteutuksessa pystyttiin käyttämään yhtä alustariippumatonta koodipohjaa, jonka oli toimittava Linux-, OSX- ja Windows-käyttöjärjestelmissä. Qt-sovelluskehyksessä kaavioiden esittämisen käyttöliittymässä mahdollisti Qt Charts -kirjasto, jonka avulla kaavioita pystyttiin integroimaan sovelluksen käyttöliittymään. Qt-sovelluskehyksessä toiminnallisuuden erottelun komponenttien välillä mahdollisti Qt model/view -arkkitehtuuri, jonka avulla pystyttiin mukauttamaan elementtien esitystä ja käyttämään erilaisia tietolähteitä.



Kuvio 9. Analytiikkatyökalu-sovelluksen luokkakaavio

Analytiikkatyökalu-sovelluksen suunnittelussa on hyödynnetty model/view -arkkitehtuuria, jonka avulla voidaan hallita tiedon ja esityksen välistä suhdetta ja erottaa toiminnallisuudet erilaisten komponenttien välille. Arkkitehtuurissa tietoa taulukon muodossa edustaa model-komponentti, jonka rajapinnan avulla view-komponentit pääsevät käsiksi tietoihin. Arkkitehtuurissa tietoa näkymässä esittää view-komponentti, jonka esittämän tiedon ei tarvitse muistuttaa model-komponentin toimittaman tiedon muotoa. Arkkitehtuurissa signal and slot -mekanismi mahdollistaa käyttöliittymässä sijaitsevien widget-elementtien tapahtumankäsittelyn, jotka ilmoittavat käyttöliittymän tapahtumista, joihin vastataan kutsumalla tiettyä toiminnallisuutta (ks. kuvio 10). Arkkitehtuurissa model-komponenttien ja view-komponenttien välisen viestinnän mahdollistaa signal and slot -mekanismi, jonka avulla model-komponentit voivat ilmoittaa tiedoissa tapahtuvista muutoksista niihin liitetyille view-komponenteille.

```
class outputView : public QDialog {
    Q_OBJECT
private slots:
    void outputChanged(int index);
};

connect(outputAxisXComboBox, SIGNAL(currentIndexChanged(int)), this,
SLOT(outputChanged(int)));
connect(outputAxisYComboBox, SIGNAL(currentIndexChanged(int)), this,
SLOT(outputChanged(int)));
```

Kuvio 10. Widget-elementtien tapahtumankäsittely

Analytiikkatyökalu-sovelluksessa toteutetut toiminnallisuudet ovat käyttöliittymän toteutus, jaetun kirjaston lataaminen, kaavion esittäminen, kaavion zoomaus, kaavion tallentaminen, tietojen lukeminen ja tietojen kirjoittaminen. Analytiikkatyökalu-sovelluksella voidaan asettaa käyttöliittymässä profiilitietoja ja mittaustietoja sekä palauttaa analyysin ulostuloja erilaisilta laitekirjastoilta erilaisten ajonaikaisten analytiikkatyökalu-kirjastojen välityksellä. Analytiikkatyökalu-sovellus varastoi analytiikkatyökalu-kirjastolta saapuvat analyysin ulostulot model-komponentteihin, minkä jälkeen view-komponentit esittävät analyysin ulostulot kaavioina sovelluksen käyttöliittymässä.

Analytiikkatyökalu-sovelluksessa ikkuna on toteutettu perimällä QMainWindow-luokan toiminnallisuudet, jotka antavat ikkunalle useita valmiiksi määritettyjä toiminnallisuuksia, kuten oletus asettelun sekä palauta ikkuna tehtäväpalkkiin-, palauta pieneksi ikkunaksi-, palauta suureksi ikkunaksi- ja sulje-toiminto. Sovelluksen käyttöliittymän asettelussa on hyödynnetty QGridLayout-luokkaa, joka mahdollistaa ruudukkoon perustuvan sisällön asettelun (ks. kuvio 11). Ruudukkoon voidaan asettaa uusia widget-elementtejä QGridLayout-luokan addWidget-jäsenfuktiolla rivin ja sarakkeen perusteella. Sovelluksen käyttöliittymässä on käytetty useita erityyppisiä widget-elementtejä, kuten kaavio-, painike-, pudotusvalikko-, syötekenttä-, tilarivi- ja valikko-elementtejä.

```
QComboBox *outputAxisXComboBox = new QComboBox;
QComboBox *outputAxisYComboBox = new QComboBox;

QGridLayout *chartSelectionGridLayout = new QGridLayout;
chartSelectionGridLayout->addWidget(outputAxisXComboBox, 0, 0);
chartSelectionGridLayout->addWidget(outputAxisYComboBox, 0, 1);

QGroupBox *chartSelectionGroupBox = new QGroupBox(tr("Output"));
chartSelectionGroupBox->setLayout(chartSelectionGridLayout);

QGridLayout *mainLayout = new QGridLayout;
mainLayout->addWidget(chartSelectionGroupBox, 0, 0);
mainLayout->addWidget(chartView, 1, 0);
setLayout(mainLayout);
```

Kuvio 11. Widget-elementtien asettelu

Analytiikkatyökalu-sovelluksessa jaettu kirjasto ladataan ajonaikaisesti QLibrary-luokan toiminnallisuudella, joka ottaa syötteeksi absoluuttisen polun ladattavaan jaettuun kirjastoon, jolla on tiedostopääte so, dynlib tai dll. Sovelluksessa tarkastetaan jaetun kirjasto lataamisen onnistuminen QLibrary-luokan load-jäsenfunktiolla. Sovelluksessa toiminnallisuus ladataan ajonaikaisesta kirjastosta QLibrary-luokan resolve-jäsenfunktiolla, joka palauttaa osoitteen ajonaikaisesta kirjastosta tuotuun toiminnallisuuteen (ks. kuvio 12). Sovelluksessa ajonaikaisesta kirjastosta tuodaan QLibrary-luokan resolve-jäsenfunktiolla createWrapper- ja deleteWrapper-funktiot, joiden avulla voidaan luoda, käyttää ja poistaa Wrapper-luokan ilmentymiä.

```
#ifdef _WIN32
#define EXPORT __declspec(dllexport)
#else
#define EXPORT
#endif

class EXPORT Wrapper {
    // ...
};

EXPORT Wrapper *createWrapper() {
    return new Wrapper();
}

EXPORT void deleteWrapper(Wrapper *wrapper) {
    delete wrapper;
}

typedef Wrapper *(*CreateWrapper)();
createWrapper = (CreateWrapper)library.resolve("createWrapper");

Wrapper *wrapper = createWrapper();
```

Kuvio 12. Dynaamisen kirjaston lataaminen

Analytiikkatyökalu-sovelluksessa kaavion esittäminen on toteutettu ChartView-luokassa perimällä QChart-luokka, joka antaa luokalle toiminnallisuuksia, kuten addAxis- ja addSeries-jäsenfunktiot, joiden avulla asetetaan akselit ja sarjat. Kaaviolle asetetaan akselit QValueAxis-luokan ilmentymällä, jonka setTitleText-jäsenfunktiolla asetetaan akselille kuvaus ja setLabelFormat asettaa akselille esitysformaatin. Viivakaaviolle asetetaan sarjat QLineSeries-luokan ilmentymällä, jonka setName-jäsenfunktiolla asetetaan sarjalle nimi. Kaavion akselit ja sarjat kiinnitetään toisiinsa QLineSeries-luokan attachAxis-jäsenfunktiolla. Kaavion arvopisteet asetetaan QVXYModelMapper-luokan ilmentymällä, jonka setXColumn- ja setYColumn-jäsenfunktiot määrittävät rivin ja sarakkeen, setSeries-jäsenfunktio määrittää sarjan ja setModel määritää model-komponentin. Model-komponenttina käytetään QStandardItemModel-luokan ilmentymiä, jotka edustavat arvopisteitä taulukon muodossa (ks. kuvio 13).

```
QValueAxis *axisX = new QValueAxis();
axisX->setTitleText(outputAxisXComboBox->currentText());
axisX->setLabelFormat("%d");
QValueAxis *axisY = new QValueAxis();
axisY->setTitleText(outputAxisYComboBox->currentText());
axisY->setLabelFormat("%d");

QLineSeries *series = new QLineSeries();
series->setName(serieName);

QHXYModelMapper *mapper = new QHXYModelMapper(this);
mapper->setXRow(outputAxisXComboBox->currentIndex());
mapper->setYRow(outputAxisYComboBox->currentIndex());
mapper->setSeries(series);
mapper->setModel(model);

QChart *chart = new QChart();
chart->setTitle(chartTitle);
chart->addAxis(axisX, Qt::AlignBottom);
chart->addAxis(axisY, Qt::AlignLeft);
series->attachAxis(axisX);
series->attachAxis(axisY);

setChart(chart);
```

Kuvio 13. Kaavion esittäminen

Analytiikkatyökalu-sovelluksessa kaavion zoomaus on toteutettu ChartView-luokassa perimällä QChartView-luokan toiminnallisuudet, jotka antavat kaaviolle valmiiksi määriteltyjä toiminnallisuuksia, kuten setRubberBand-jäsenfunktio sekä viewportEvent-, mouseMoveEvent-, mousePressEvent- ja mouseReleaseEvent-tapahtumankäsittelijät. Zoomaus toteutetaan hiirelle ja kosketukselle QChartView-luokalla, jonka setRubberBand-jäsenfunktiolle annetaan argumenttina RectangleRubberBand-enumeraatio, joka mahdollistaa zoomauksen pysty- ja vaakasuoraan. Zoomaus toteutetaan kosketukselle lipulla. Lippu asetetaan päälle, kun viewportEvent-tapahtumankäsittelijä havaitsee kosketus-tapahtuman alkamisen. Oletustoiminnallisuudet estetään viewportEvent-tapahtumankäsittelijän avulla mouseMoveEvent- ja mousePressEvent-tapahtumankäsittelijöissä. Lippu asetetaan pois päältä, kun mouseReleaseEvent-tapahtumankäsittelijä havaitsee kosketus-tapahtuman päättymisen.

Analytiikkatyökalu-sovelluksessa kaavion tallentaminen on toteutettu ChartView-luokassa perimällä QChartView-luokan toiminnallisuudet, jotka lisäävät kaaviolle valmiiksi määritettyä toiminnallisuutta QWidget-luokasta, kuten grab-jäsenfunktion, jonka avulla kaavio voidaan muuntaa QPixmap-luokan ilmentymäksi. Kaavio tallennetaan kuvaksi QPixmap-luokan save-jäsenfunktiolla, joka ottaa argumentteina

tiedostosijainnin ja kuvatiedostoformaatin (ks. kuvio 14). Tuettuja kuvatiedostoformaatteja kuvan tallentamisessa ovat BMP-, JPG-, JPEG-, PNG-, PPM-, XBM- ja XPM-formaatit.

```
QPixmap CustomChartView::toQPixmap() {
    return grab();
}

QPixmap pixmap = customChartView->toQPixmap();
bool status = pixmap.save(fileName, "PNG");
```

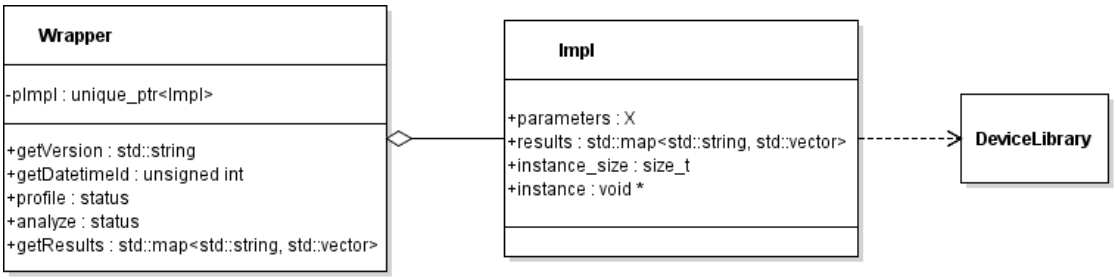
Kuvio 14. Kaavion tallentaminen

Analytiikkatyökalu-sovelluksessa profiilitiedot luetaan CSV-tiedostosta ProfileView-luokan openFile-jäsenfunktion toiminnallisuudella, jonka avulla valitaan, avataan ja luetaan tiedosto. Tiedosto valitaan QFileDialog-luokan getOpenFileName-jäsenfunktiolla, jonka avulla voidaan selata tiedostojärjestelmää sekä valita tiedosto. Tiedosto avataan QFile-luokan rakentajalla, jolle asetetaan argumenttina avattavan tiedoston polku. Tiedosto luetaan QTextStream-luokalla, jonka rakentajalle asetetaan argumenttina avattua tiedostoa edustava QFile-luokan ilmentymä ja readLine-jäsenfunktiolla luetaan tiedot rivi kerrallaan tiedostosta, jossa tekstikentät on eroteltu toisistaan puolipisteellä ja rivit rivinvaihdoilla. Luetut tiedot kirjoitetaan QStandardItemModel-luokan ilmentymään, jonka insertRow-jäsenfunktio kirjoittaa tiedot.

Analytiikkatyökalu-sovelluksessa profiilitiedot kirjoitetaan CSV-tiedostoon ProfileView-luokan saveFile- jäsenfunktion toiminnallisuudella, jonka avulla luodaan, avataan ja kirjoitetaan tiedosto. Tiedosto luodaan QFileDialog-luokan getSaveFileName-jäsenfunktiolla, jonka avulla voidaan selata tiedostojärjestelmää sekä luoda tiedosto. Tiedosto avataan QFile-luokan rakentajalla, jolle asetetaan argumenttina avattavan tiedoston polku. Tiedosto kirjoitetaan QTextStream-luokalla, jonka rakentajalle asetetaan argumenttina avattua tiedostoa edustava QFile-luokan ilmentymä ja ulostulo-operaattorilla kirjoitetaan tiedot rivi kerrallaan tiedostoon, jossa tekstikentät on eroteltu toisistaan puolipisteellä ja rivit rivinvaihdoilla. Kirjoitetut tiedot luetaan QStandardItemModel-luokan ilmentymän data-jäsenfunktiolla.

4.2 Analytiikkatyökalu-kirjasto

Analytiikkatyökalu-kirjaston tehtävänä oli varmistaa yhteensopivuus analytiikkatyökalu-sovellukselle ja laitekirjastolle. Analytiikkatyökalu-kirjaston toteutuksessa käytettiin C++-ohjelmointikielen STL-kirjastoa sekä suunnittelussa hyödynnettiin Bridge-suunnittelumallia. Analytiikkatyökalu-kirjaston luokkakaavio on esitetty kuviossa 15. STL-kirjasto mahdollisti optimoidummat ja yleisemmät tietorakenteet kuin Qt-sovelluskehyksen tarjoamat vastaavat tietorakenteet. Kirjastosta pystyttiin poistamaan Qt-sovelluskehys riippuvuus kokonaan, koska haluttiin välttää Qt-sovelluskehys riippuvuus kirjaston käännösvaiheesta, kun käännöksen tekevällä laitteella ei ollut välttämättä Qt-sovelluskehystä asennettuna. Bridge-suunnittelumallin tarkoituksena oli erottaa kirjaston abstraktio sekä toteutus siten, että nämä kaksi voivat vaihdella itsenäisesti, koska haluttiin välttää pysyvä sitoutuminen abstraktion ja toteutuksen välillä, kun erilaisia kirjastoja oli pystyttävä valitsemaan ja vaihtamaan ajonaikaisesti.



Kuvio 15. Analytiikkatyökalu-kirjaston luokkakaavio

Analytiikkatyökalu-kirjaston suunnittelussa on hyödynnetty Bridge-suunnittelumallia, jonka avulla voidaan toteuttaa julkinen abstraktio ja useita yksityisiä toteutuksia, mikä mahdollistaa kirjaston yhteensopivuuden julkisen abstraktion ja erilaisten yksityisten toteutuksien välillä (ks. kuvio 16). Analytiikkatyökalu-kirjaston julkinen abstraktio on aina samanlainen ja yhteensopiva, koska muutos julkisessa abstraktiossa aiheuttaisi yhteensopivuus ongelman kirjastoa käyttävälle sovellukselle, minkä seurauksena kirjastoa käyttävä sovellus pitäisi tehdä yhteensopivaksi ja kääntää uudelleen. Analytiikkatyökalu-kirjaston yksityinen toteutus ei ole aina samanlainen,

koska yksityinen toteutus riippuu laitekirjaston tukemasta toiminnallisuudesta, jota vasten kirjasto on käännetty. Yksityisen toteutuksen eroavaisuuden sovelluksessa mahdollistavat esikääntäjät, jotka lisäävät tai rajaavat kirjaston toiminnallisuutta ennen käännöstä.

```
// File: wrapper.h
class EXPORT Wrapper {
public:
    virtual setProfile(std::map<std::string, double> &settings);
private:
    struct Impl;
    std::unique_ptr<Impl> pImpl;
};

// File: feature.h
struct Features {
#ifdef FEATURE_A
    double variable;
#endif
};

// File: wrapper.cpp
#include <feature.h>
struct Wrapper::Impl {
    Features features;
};

Wrapper::setProfile(std::map<std::string, double> &settings) {
#ifdef FEATURE_A
    pImpl->features.variable = settings["variable"];
#endif
}
```

Kuvio 16. Kirjaston yhteensopivuuden muodostaminen

Analytiikkatyökalu-kirjastossa toteutetut toiminnallisuudet ovat profiilitietojen asettaminen, analyysin suorittaminen ja ulostulojen palauttaminen, joiden avulla ulkopuolinen sovellus voi käyttää kirjastoa vaihtamaan tietoja yhteensopivasti laitekirjaston kanssa. Analytiikkatyökalu-kirjasto ja laitekirjasto on linkitetty staattisesti, minkä tarkoituksena on yhdistää ja muuntaa kirjastot käännösvaiheessa yhdeksi ajonaikaiseksi kirjastoksi. Analytiikkatyökalu-kirjastoa voidaan käyttää turvallisesti ajonaikaisena eli dynaamisena tai jaettuna kirjastona, koska kirjaston binäärinen yhteensopivuus on varmistettu.

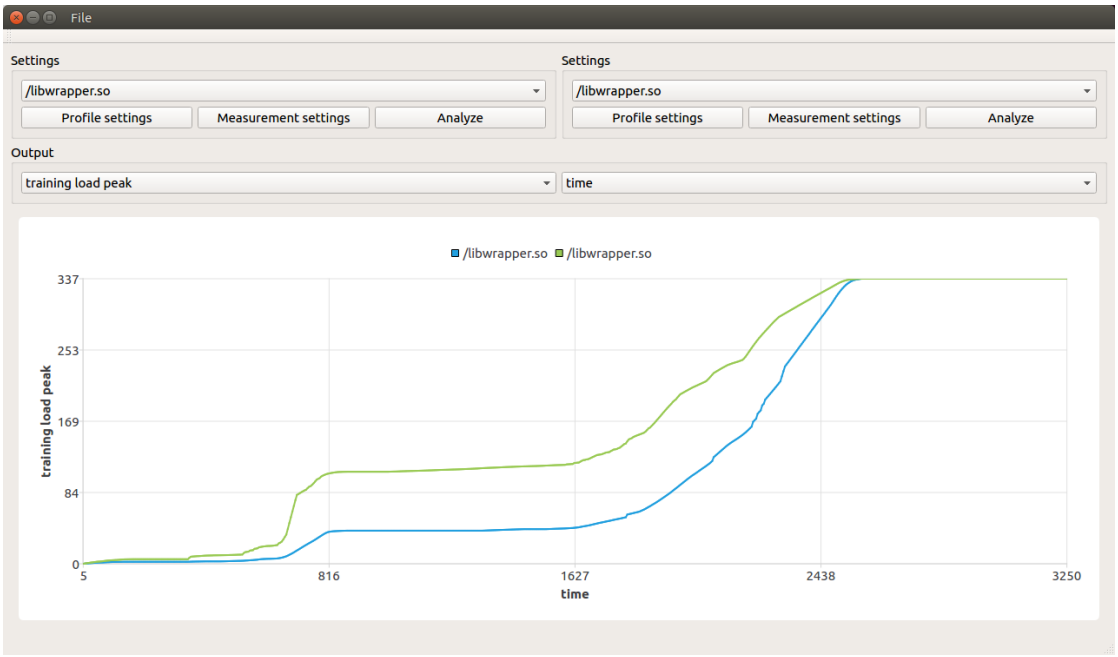
Analytiikkatyökalu-kirjastossa profiilitiedot asetetaan Wrapper-luokalla, jonka profile-jäsenfunktio ottaa argumenttina STL-kirjaston map-tietorakenteen, joka sisältää profiilitietoja, kuten ikä, paino, pituus, sukupuoli jne. Profiilitiedot palautetaan tietorakenteesta map-luokan find-jäsenfunktiolla, joka etsii profiilitiedot tietorakenteesta avaimen perusteella. Tietorakenteen find-jäsenfunktio sallii tilanteen, jossa avainta ei ole olemassa. Profiilitiedot asetetaan Wrapper-luokan yksityisessä toteutuksessa sijaitsevaan tietorakenteeseen, jonka toteutus ei ole aina samanlainen, koska tietorakenteen toteutus riippuu laitekirjaston tukemista ominaisuuksista. Profiilitietojen asettamisen tietorakenteeseen mahdollistavat esikääntäjät, jotka lisäävät tuetut tai poistavat ei tuetut toiminnallisuudet ennen käännösvaihetta.

Analytiikkatyökalu-kirjastossa analyysi suoritetaan Wrapper-luokalla, jonka analyze-jäsenfunktio ottaa argumenttina STL-kirjaston map-tietorakenteen, joka sisältää vector-tietorakenteita, jotka sisältävät mittaustietoja. Mittaustiedot palautetaan tietorakenteesta map-luokan find-jäsenfunktiolla, joka etsii mittaustiedot tietorakenteesta avaimen perusteella. Tietorakenteen find-jäsenfunktio sallii tilanteen, jossa avainta ei ole olemassa. Mittaustiedot asetetaan paikallisiin vector-tietorakenteisiin, joista syötetään mittaustietoa analyyseille. Analyysejä suoritetaan viiden sekunnin aikaväleillä profiilitietojen ja mittaustietojen pohjalta, minkä aikana asetetaan useita laitekirjaston tukemia analyysin ulostuloja useisiin vector-tietorakenteisiin, jotka koostetaan lopuksi yhdeksi map-tietorakenteeksi. Ulostulojen asettamisen tietorakenteisiin mahdollistavat esikääntäjät, jotka lisäävät tuetut tai poistavat ei tuetut toiminnallisuudet ennen käännösvaihetta.

Analytiikkatyökalu-kirjastossa ulostulot palautetaan Wrapper-luokalla, jonka getResults-jäsenfunktio palauttaa STL-kirjaston map-tietorakenteen paluuarvona, joka sisältää vector-tietorakenteita, jotka sisältävät analyytiikkakirjaston ulostuloja. Analyysin aikana asetetaan useita laitekirjaston tukemia analyysin ulostuloja useisiin vector-tietorakenteisiin, jotka koostetaan lopuksi yhdeksi map-tietorakenteeksi. Ulostulot palautetaan map-tietorakenteessa, jonka sisältö ei ole aina samanlainen, koska tietorakenteen sisältö riippuu laitekirjaston tukemista ominaisuuksista. Ulostulojen palauttamisen tietorakenteessa mahdollistavat esikääntäjät, jotka lisäävät tuetut tai poistavat ei tuetut toiminnallisuudet ennen käännösvaihetta.

5 Tulokset

Opinnäytetyön tuloksena oli analytiikkatyökalu, joka koostui analytiikkatyökalu-sovelluksesta ja analytiikkatyökalu-kirjastosta. Analytiikkatyökalu-sovellus oli alustariippumaton sovellus, jonka tehtävänä oli asettaa mittaustiedot ja profiilitiedot analytiikkatyökalu-kirjastolle sekä esittää kaavioita analytiikkatyökalu-kirjaston palauttamista analyysien ulostuloista (ks. kuvio 17). Analytiikkatyökalu-kirjasto oli yhteensopivuuden muodostava ajonaikainen kirjasto, jonka tehtävänä oli asettaa analyysien tarvitsemat profiilitiedot ja mittaustiedot laitekirjastolle analysoitavaksi sekä palauttaa analyysien tulokset laitekirjastolta. Analytiikkatyökalu-kirjaston avulla voidaan laitekirjastolle asettaa profiilitiedot ja mittaustiedot sekä suorittaa analyysejä. Analytiikkatyökalun kaikki näkymät on esitetty liitteessä 2.



Kuvio 17. Analytiikka-sovelluksen päänäkymä

Analytiikkatyökalu-sovelluksen tehtävänä oli esittää kaavioita alustariippumattomasti analytiikkatyökalu-kirjaston ulostuloista. Analytiikkatyökalu-sovelluksen avulla voidaan visuaalisesti analysoida ja verrata erilaisien profiilitietojen ja mittaustietojen vaikutuksia kirjaston ulostuloihin tai erilaisten kirjastojen tuottamia ulostuloja. Ana-

lytiikkatyökalu-sovelluksessa profiilitiedot asetetaan sovelluksen käyttöliittymässä sijaitsevasta profiilinäkymästä, jonka avulla profiilitiedot asetetaan käyttöliittymässä käsin tai ladataan ulkoisesta CSV-tiedostosta. Analytiikkatyökalu-sovelluksessa mittaustiedot asetetaan sovelluksen käyttöliittymässä sijaitsevasta mittausnäkymästä, jonka avulla mittaustiedot asetetaan C++-lähdekooditiedostosta. Analytiikkatyökalu-sovelluksessa käyttöliittymässä sijaitsevia kaavioita voidaan zoomata ja kaavio voidaan tallentaa PNG-tiedostoformaatissa käyttäjän valitsemaan tiedostosijaintiin.

Analytiikkatyökalu-kirjaston tehtävänä oli muodostaa ajonaikainen yhteensopivuus analytiikkatyökalu-sovellukselle ja laitekirjastolle. Analytiikkatyökalu-kirjaston avulla voidaan laitekirjastolle asettaa profiilitiedot, mittaustiedot ja suorittaa analyysejä. Analytiikkatyökalu-kirjastossa profiilitiedot asetetaan laitekirjastolle analytiikkatyökalu-sovelluksen käyttöliittymän profiilinäkymästä, jonka avulla profiilitietojen asettaminen voidaan tehdä käsin tai lataamalla CSV-tiedostosta. Analytiikkatyökalu-kirjastossa mittaustiedot asetetaan laitekirjastolle analytiikkatyökalu-sovelluksen käyttöliittymän mittausnäkymästä, jonka avulla mittaustiedot asetetaan C++-lähdekooditiedostosta. Analytiikkatyökalu-kirjastossa analyysi suoritetaan asetettujen profiilitietojen ja mittaustietojen pohjalta, joiden avulla suoritetaan useita analyysejä viiden sekunnin aikavälein suorittamalla laitekirjaston toiminnallisuutta.

Opinnäytetyön tuloksena oli analytiikkatyökalu, joka toteutti osan sovelluksen vaatimusmäärittelyssä kuvatuista vaatimuksista, jotka on toteutettu kokonaan tai osittain toimeksiantajan vaatimusmäärittelyn mukaisesti. Analytiikkatyökalun vaatimusmäärittelyssä määriteltiin toteutettavalle sovellukselle asetetut toiminnalliset ja ei-toiminnalliset vaatimukset, joista sovellukselle toteutettiin perustavanlaatuiset ja tärkeimmät toiminnallisuudet sovelluksen kehityksen ja toiminnan kannalta, jotka vaativat vielä muutoksia ennen kuin toiminnallisuuksien toteutukset ovat lopullisessa muodossa. Analytiikkatyökalussa toteutetut toiminnalliset vaatimukset on kuvattu taulukossa 2 ja ei-toiminnaliset vaatimukset taulukossa 3. Analytiikkatyökalun vaatimusmäärittelyssä määriteltiin sovellukselle vaatimuksia, jotka olivat rajattu opinnäytetyön toteutuksen ulkopuolelle, kuten mittaustietojen lukemiseen tarkoitetut menetelmät FBE-, FIT- ja SDF-tiedostoformaateista.

Taulukko 2. Analytiikkatyökalussa toteutetut toiminnalliset vaatimukset

Vaatimuksen kuvaus
Kaavion tuottaminen kaikista laitekirjaston ulostuloista
Kaavion zoomaus
Profiilitietojen kirjoittaminen CSV-tiedostoon
Profiilitietojen asettaminen sallituilla tiedoilla
Analyysin ulostulojen tallentaminen CSV-tiedostoon

Taulukko 3. Analytiikkatyökalussa toteutetut ei-toiminnalliset vaatimukset

Vaatimuksen kuvaus
Analytiikkatyökalun siirrettävyys Linux-, OSX- ja Windows-käyttöjärjestelmille
Analytiikkatyökalun yhteensopivuus erilaisille laitekirjastoille

Opinnäytetyön tuloksena ratkaistiin opinnäytetyölle asetetut seuraavat kehittämistehtävät: Miten toteutetaan alustariippumattomuus työpöytäsovelluksessa? Miten toteutetaan yhteensopivuus sovelluksen ja erilaisten kirjastojen välillä? Opinnäytetyössä alustariippumattomuus analytiikkatyökalu-sovelluksessa ratkaistiin käyttämällä Qt-sovelluskehystä, joka mahdollisti GUI-työpöytäsovelluksen toteuttamisen alustariippumattomasti, jonka avulla analytiikkatyökalu-sovelluksen toteutuksessa pystyttiin käyttämään yhtä alustariippumatonta koodipohjaa, joka oli käännettävissä Linux- ja Windows-käyttöjärjestelmissä. Opinnäytetyössä yhteensopivuus sovelluksen ja erilaisten kirjastojen välillä ratkaistiin käyttämällä Bridge-suunnittelumallia, joka mahdollisti analytiikkatyökalu-kirjaston ajonaikaisen yhteensopivuuden samanlaisen julkisen abstraktion ja erilaisten yksityisten toteutuksien välillä. Yksityisen toteutuksen eroavaisuuden mahdollistivat esikääntäjät, jotka käsittelivät toteutuksen ennen käännösprosessia, joiden avulla kirjaston toiminnallisuutta lisättiin tai rajattiin.

6 Pohdinta

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa laitekirjaston analysointiin analysointityökalu, jonka avulla voidaan visuaalisesti analysoida ja verrata erilaisten profiilitietojen ja mittaustietojen vaikutuksia laitekirjaston ulostuloihin. Opinnäytetyön rajaus huomioiden asetettuun tavoitteeseen päästiin, koska opinnäytetyön tuloksena oli analytiikkatyökalu, joka toteutti opinnäytetyölle asetetun tavoitteen ja osan analytiikkatyökalun vaatimusmäärittelyn vaatimuksista. Analytiikkatyökalu-sovelluksella oli mahdollista esittää kaavioita analytiikkatyökalu-kirjaston ulostuloista. Analytiikkatyökalu-kirjastolla oli mahdollista muodostaa ajonaikainen yhteensopivuus laitekirjastolle. Analytiikkatyökalu ei ole vielä valmis, koska se ei täytä kaikkia vaatimusmäärittelyn vaatimuksia.

Opinnäytetyön menetelmät määritteli analytiikkatyökalun vaatimusmäärittely, jonka toiminnallisten ja ei-toiminnallisten vaatimusten pohjalta valittiin toteutuksessa käytettävät menetelmät. Opinnäytetyön menetelmien kannalta keskeisimmiksi vaatimusmäärittelyn vaatimuksiksi muodostui analytiikkatyökalun siirrettävyys eri käyttöjärjestelmille, yhteensopivuus eri laitekirjastoille ja kaavioiden esittäminen laitekirjaston ulostuloista. Analytiikkatyökalu-sovelluksen toteutuksessa käytettäväksi menetelmäksi valittiin Qt-sovelluskehys, koska sovelluskehyksen avulla oli mahdollista toteuttaa sovelluksen siirrettävyys vaadituille käyttöjärjestelmille ja Qt Chart -moduulin avulla oli mahdollista sovelluksessa vaadittujen kaavioiden esittäminen. Analytiikkatyökalu-kirjaston toteutuksessa käytettäväksi menetelmiksi valittiin Bridge-suunnittelumalli, koska suunnittelumallin avulla oli mahdollista muodostaa yhteensopivuus erilaisille laitekirjastoille.

Opinnäytetyössä onnistuttiin kehittämään analytiikkatyökalusta toimiva kokonaisuus, joka oli alustariippumaton osalle vaadituista käyttöjärjestelmistä ja yhteensopiva erilaisille laitekirjastoille. Analytiikkatyökalu-sovelluksen alustariippumattomuus saavutettiin Qt-sovelluskehyksellä, joka oli käännettävissä Linux- ja Windows-käyttöjärjestelmissä. Analytiikkatyökalun ylläpito on yksinkertaista, koska aina kun laitekirjastolle lisätään uusi ulostulo niin analytiikkatyökalu-kirjastolle lisätään uusi ulostulo. Analytiikkatyökalu-sovellusta ei tarvitse päivittää, koska analytiikkatyökalu-sovellus esittää analytiikkatyökalu-kirjaston palauttamat ulostulot dynaamisesti. Analytiikkatyökalu-

kirjaston yhteensopivuus saavutettiin esikääntäjillä ja Bridge-suunnittelumallilla, joiden avulla kirjastosta saatiin eteen- ja taaksepäin yhteensopiva erilaisille laitekirjastoille. Analytiikkatyökalu-kirjaston yhteensopivuus on voimassa niin kauan kuin laitekirjaston rakenne on sama ja tuetut ominaisuudet on määritelty esikääntäjillä.

Opinnäytetyössä ei onnistuttu toteuttamaan kaikkia analytiikkatyökalun vaatimusmäärittelyssä kuvattuja vaatimuksia. Analytiikkatyökalussa on vielä toteutettavana analyysien suorittamiseen liittyvää toiminnalllisuutta ja mittaustietojen lukemiseen liittyvää toiminnallisuutta FBE-, FIT- ja SDF-tiedostoformaateista. Analytiikkatyökalun käyttöliittymä on toteutettu kokonaan toiminnallisuuksien testaamista varten eikä se vastaa sovelluksen lopullista käyttöliittymää. Lopullisen käyttöliittymän toteutus voidaan aloittaa, kun käyttöliittymälle saadaan suunnitelmat. Analytiikkatyökalu vaatii ennen käyttöönottoa perusteellisen testaamisen, koska kyseessä on työkalu, jonka avulla analysoidaan laitekirjaston ulostuloja ja laitekirjaston toimintaa. Testaamisessa on huomioitava siirrettävyyden tuomat ongelmat ja erityisesti kiinnitettävä huomioita muuttuja tyyppeihin, joiden toteutus riippuu laitealustasta.

Opinnäytetyöprosessi sujui pääsääntöisesti hyvin, koska opinnäytetyö toteutettiin ennestään tutulla C++-ohjelmointikielellä ja työpöytäsovellusten toteuttaminen oli ennestään tuttua Windows-käyttöjärjestelmän natiivilla menetelmällä. Opinnäytetyö prosessin suurimmat ongelmat olivat uudet menetelmät ja aikataulu. Opinnäytetyössä käytetyistä menetelmistä ennestään tuntemattomia menetelmiä oli Qt-sovelluskehys, Qt Creator ja Qmake. Ongelmatilanteissa Qt-sovellukehys kohtaisesta tiedonhausta internetistä ei ollut hyötyä eri keskustelusivustoilta, koska tietoa oli yllättävän vähän saatavilla. Tiedonhaussa turvauduttiin lähes poikkeuksetta internetissä saatavilla olevaan Qt-dokumentaatioon. Opinäytetyössä aikataulu osoittautui liian tiukaksi, koska uusia menetelmiä oli useita ja tehtävään nähden opinnäytetyön tekeminen aloitettiin liian myöhään. Aikataulun takia opinnäytetyön raportointi jäi osittain pinnalliseksi.

Analytiikkatyökalun kehittämisideoina olisi toteuttaa pikatoiminnot laitekirjaston ulostulojen analysointiin. Pikatoimintojen tarkoituksena olisi mahdollistaa nopea vertailu kahden tai useamman laitekirjaston ulostulojen välillä, jonka voisi toteuttaa analytiikkatyökalun käyttöliittymä- ja komentorivi-laajennoksena. Käyttöliittymässä ja

komentorivillä ulostuloja voisi suodattaa pikatoiminnoilla, kuten näytä kaikki, eroavat ja samanlaiset ulostulot. Pikatoimintojen tavoitteena olisi lyhentää laitekirjaston ulostulojen analysointia, kun tehdään asiakaspaketteja, koska analyysin suorittamiseen ja analysointiin nykyisellä menetelmällä kuluu aikaa keskimäärin tunti. Pikatoiminnot voisi liittää CI/CD-ketjuun, jonka avulla vertailu suoritetaan joka kerta kun katselmointiin lisätään koodia.

Lähteet

About Qt. N.d. Qt Group Oyj:n internetsivut. Viitattu 6.11.2018. https://wiki.qt.io/About_Qt.

Blanchette, J. & Summerfield, M. 2008. C++ GUI Programming withQt 4, Second Edition. Westford Massachusetts: Prentice Hall.

Building and Running. N.d. Qt Group Oyj:n internetsivut. Viitattu 11.11.2018. http://doc.qt.io/qtcreator/creator-building-running.html.

Coding. N.d. Qt Group Oyj:n internetsivut. Viitattu 11.11.2018. http://doc.qt.io/qtcreator/creator-coding.html.

Container Classes. N.d. Qt Group Oyj:n internetsivut. Viitattu 1.12.2018. https://doc.qt.io/qt-5.11/containers.html.

Delegate Classes. N.d. Qt Group Oyj:n internetsivut. Viitattu 12.11.2018. http://doc.qt.io/qt-5/model-view-programming.html#delegate-classes.

Designing User Interfaces. N.d. Qt Group Oyj:n internetsivut. Viitattu 11.11.2018. http://doc.qt.io/qtcreator/creator-design-mode.html.

Fysiologia. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 25.11.2018. https://www.firstbeat.com/fi/fysiologia/.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

IDE. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. https://wiki.qt.io/About_Qt.

IDE Overview. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. http://doc.qt.io/qtcreator/creator-overview.html.

Input/Output and Networking. N.d. Qt Group Oyj:n internetsivut. Viitattu 1.12.2018. https://doc.qt.io/qt-5.11/io.html.

Kuluttajat. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 25.11.2018. https://www.firstbeat.com/fi/kuluttaja-tuotteet/.

Model Classes. N.d. Qt Group Oyj:n internetsivut. Viitattu 12.11.2018. http://doc.qt.io/qt-5/model-view-programming.html#model-classes.

Model/View Programming. N.d. Qt Group Oyj:n internetsivut. Viitattu 12.11.2018. http://doc.qt.io/qt-5/model-view-programming.html.

Paakki, J. 2011. Ohjelmistojen vaatimusmäärittely. Viitattu 3.11.2018.

https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf.

qmake Manual. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. http://doc.qt.io/qt-5/qmake-manual.html.

Qt Charts. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. https://doc.qt.io/qt-5.11/qtcharts-index.html.

Qt Charts Overview. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. https://doc.qt.io/qt-5.11/qtcharts-overview.html.

Qt Core. N.d. Qt Group Oyj:n internetsivut. Viitattu 1.12.2018. https://doc.qt.io/qt-5.11/qtcore-index.html.

Qt Essentials. N.d. Qt Group Oyj:n internetsivut. Viitattu 2.12.2018. https://doc.qt.io/qt-5.11/qtmodules.html.

Qt Licensing. N.d. Qt Group Oyj:n internetsivut. Viitattu 6.11.2018. http://doc.qt.io/qt-5/licensing.html.

Qt Widgets. N.d. Qt Group Oyj:n internetsivut. Viitattu 10.11.2018. https://doc.qt.io/qt-5.11/qtwidgets-index.html.

Signals & Slots. N.d. Qt Group Oyj:n internetsivut. Viitattu 9.11.2018. http://doc.qt.io/qt-5/signalsandslots.html.

Sommerville, I. 2010. Software Engineering. Ninth Edition. Pearson Education.

Tarinamme. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 16.10.2018. https://www.firstbeat.com/fi/yritys/tarina/.

Testing. N.d. Qt Group Oyj:n internetsivut. Viitattu 11.11.2018. http://doc.qt.io/qtc-reator/creator-testing.html.

The Meta-Object System. N.d. Qt Group Oyj:n internetsivut. Viitattu 7.11.2018. http://doc.qt.io/qt-5/metaobjects.html.

View Classes. N.d. Qt Group Oyj:n internetsivut. Viitattu 12.11.2018. http://doc.qt.io/qt-5/model-view-programming.html#view-classes.

Yritys. N.d. Firstbeat Technologies Oy:n internetsivut. Viitattu 27.10.2018. https://www.firstbeat.fi/fi/yritys/.

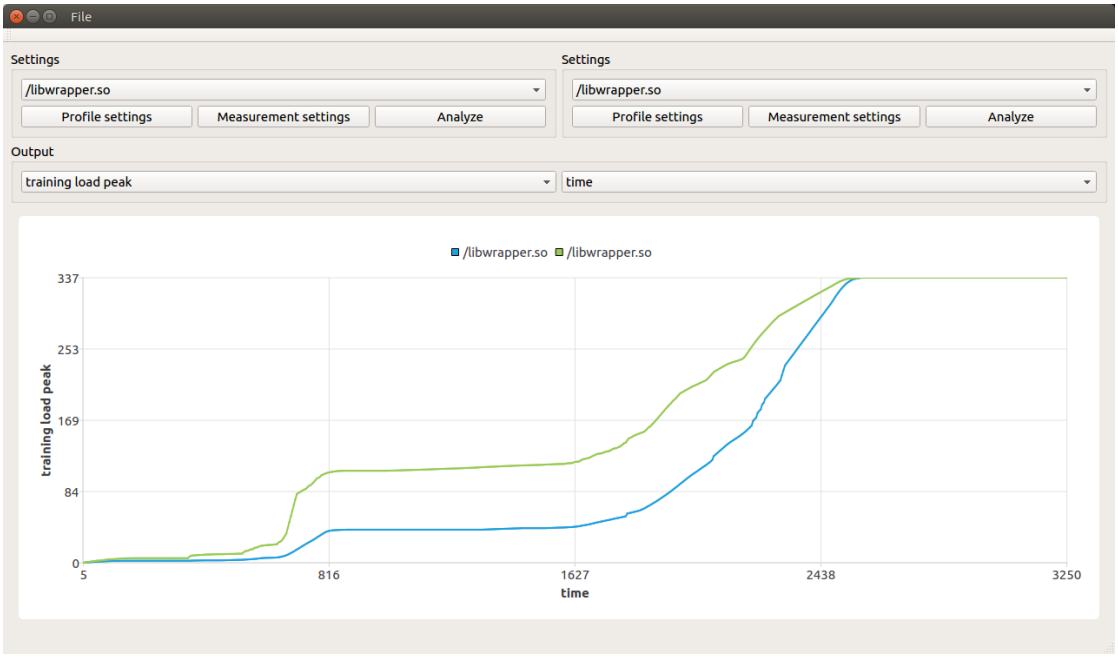
Liitteet

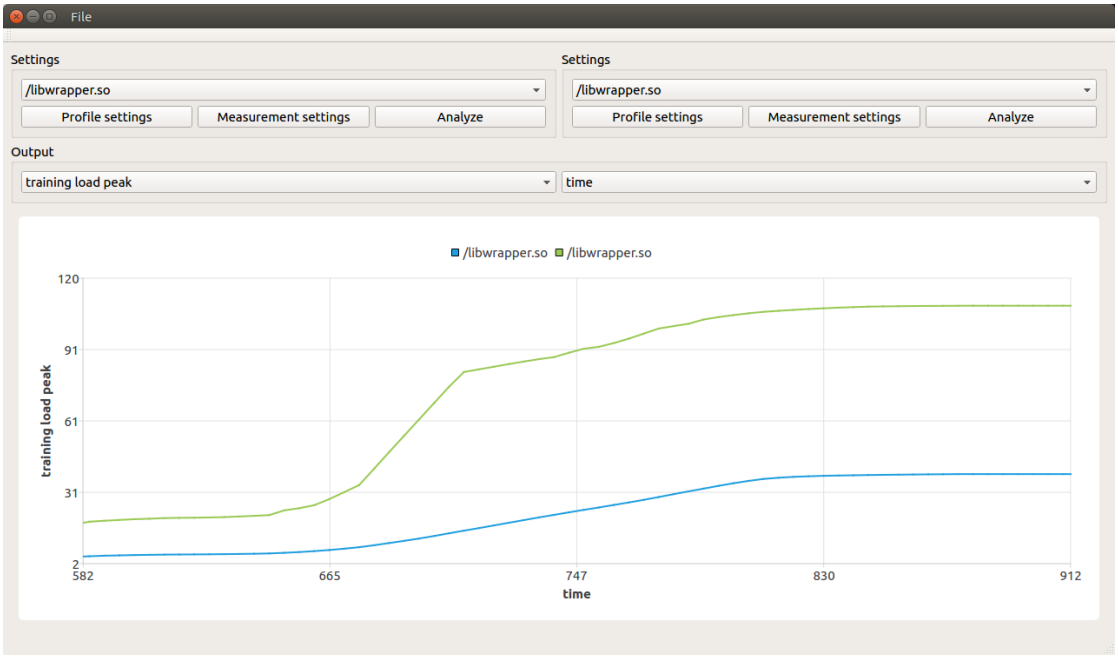
Liite 1. Analytiikkatyökalun vaatimusmäärittely

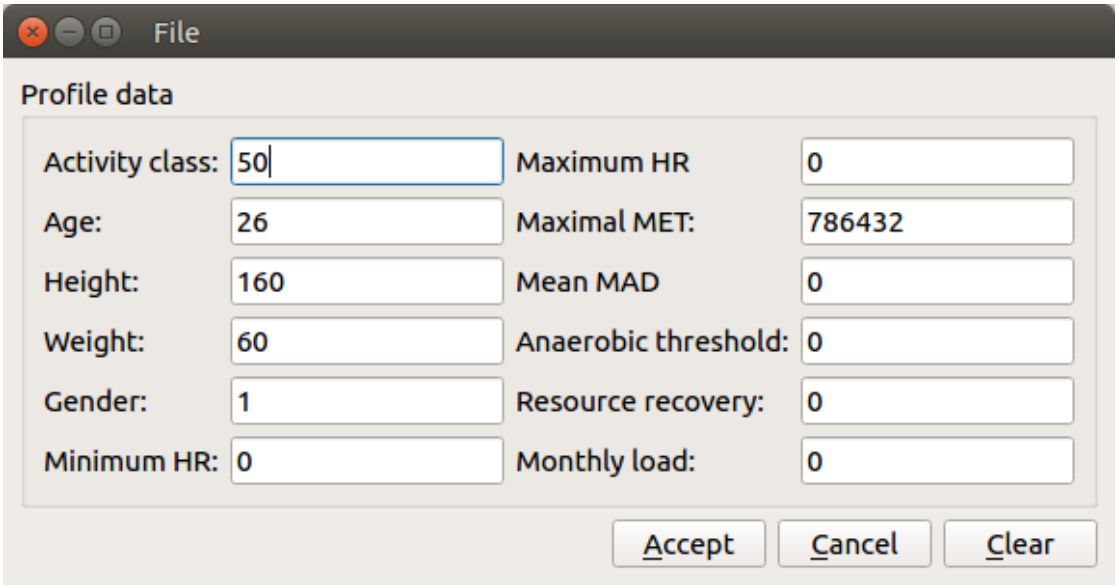
Requirement number	Requirement	Priority	Description	Done?
	Requirement short description		Define what the requirement is in detail	✔
1	Operational environment		FAT 2.0 must be able to be used at least in Windows, Mac and Linux	
2	FIT format compatibility		FAT 2.0 must be able to take measurement data from FIT file. From FIT file program is able to take into account the following information: - Profile parameters - measurement date - HR - B2B - Speed - Watts - Step rate - Altitude - Temperature - Humidity - HR quality	
3	SDF format compatibility		FAT 2.0 must be able to take measurement data form SDF files. If SDF file is paired with ACC data in CSV file format created by Firstbeat tools, FAT automatically takes the ACC data also. Also measurement date is read from SDF.	
4	FBE format compatibility		FAT 2.0 is able to take measurement data from FBE file. From FBE following info is read: - Profile info - Measurement date - B2B - Speed - Altitude	
5	ETE library options		FAT 2.0 needs to have a selection option for ETE library versions. Different library versions are set for FAT one way or another. This can be e.g. In a same way as with FAT 1.0, where user needed to simply save the ETE build into certain folder within the FAT program folder and FAT automatically then added the ETE version into the list of available ETE versions. In FAT 2.0 tool, program shows the branch name of the ETE and the library version.	
6	Reading the profile parameters from file		FAT 2.0 reads the profile parameters automatically from selected input file (in case of FIT and FBE) and pre-sets the parameters from the file as profile parameters to be used for the analysis. Parameters are modifiable after they have been read from the file.	
7	Profile parameters - available parameters filtering		FAT 2.0 allows user to set only the parameters which can be set based on the selected ETE library (Req #5).	
8	Profile parameters setting - allowed range		FAT 2.0 checks the set profile parameters and allows the user to set only valid parameters. If user tries to set an invalid parameter, an error is returned of what input is invalid and what is the allowed range.	
9	Analysis options - choosing analysis parameters/inputs		In FAT 2.0 user is able to modify following analysis parameters: - Set walking fitness test ON/OFF - Set altisource to GPS/BARO - Set the user state - Select which input data to use (B2B, HR, speed, watts, altitude, step rate, temperature, humidity, hr quality). By default all are used.	
10	Analysis options - run updating parameters		In case user selects at the same time more then one file for analysis, these measurements can be set to be analyzed in a row by updating the personalization parameters between the measurements. This simulates how the data is analyzed in a device where e.g. maximal met gets updated from the measurement to profile settings and from there to next analysis.	
11	Analysis options - internal variables to analysis results		FAT 2.0 allows ETE internal variables to be saved to results.	
12				

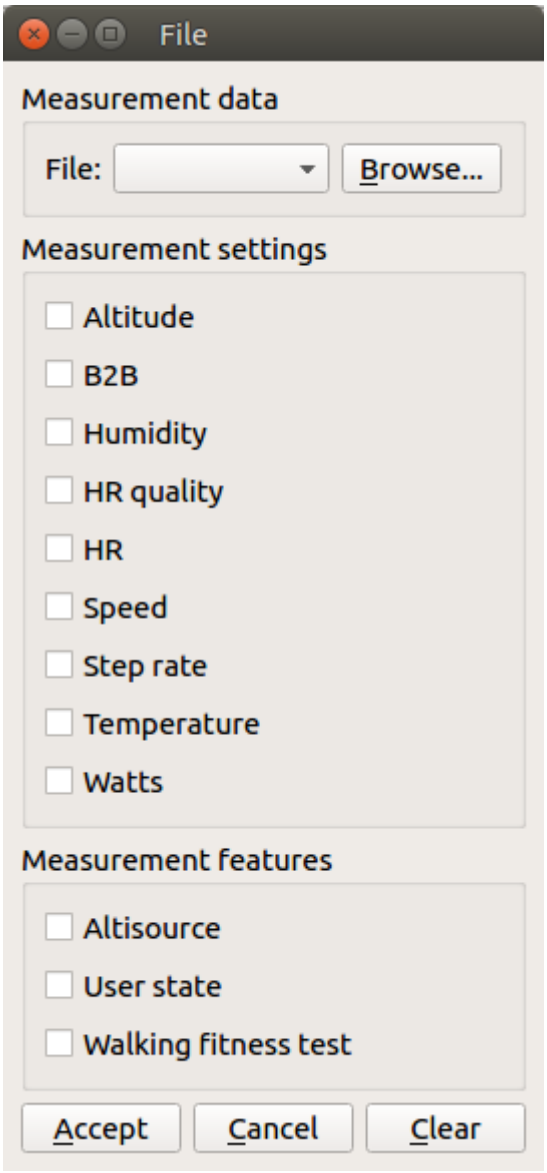
Requirement number	Requirement	Priority	Description	Done?
1	Saving results into CSV	Must	FAT 2.0 creates a CSV file of the analysis results. This file contains all the library outputs printed in a result row per 5 second timing. See example from Dropbox (Firstbeat)/Consumer/Projects/FAT 2.0/Example-results-file.csv	
2	Plotting the results		FAT 2.0 is able to plot all the ETE results. User is able to select at least two parameters to be plotted into the same graph. X-axis of the graph is shown as minutes or time of day (start time taken from the input data file)	
3	Plot zoom		In FAT 2.0 user is able to zoom into the results plot.	
4	Analysis summary results		FAT 2.0 is able to make an CSV of profile parameters used in the analysis if updating parameters selection is made (requirement no. 10 at analysis sheet). CSV has line per measurement where all the profile parameters used in the analysis is presented. Number of parameters depends what parameters are to be set in the selected ETE version. Parameters in the line represents what was given to ETE for the given measurement analysis with set_parameters call.	

Liite 2. Analytiikkatyökalu-sovelluksen käyttöliittymä






File
Profile data
Activity class: 50
Maximum HR 0
Age: 26
Maximal MET: 786432
Height: 160
Mean MAD 0
Weight: 60
Anaerobic threshold: 0
Gender: 1
Resource recovery: 0
Minimum HR: 0
Monthly load: 0
Accept
Cancel
Clear


File
Measurement data
File:
Browse...
Measurement settings
Altitude
B2B
Humidity
HR quality
HR
Speed
Step rate
Temperature
Watts
Measurement features
Altisource
User state
Walking fitness test
Accept
Cancel
Clear